ARMY RESEARCH LABORATORY

Predicting Coalescence of Blast Waves From Sequentially Exploding Ammunition Stacks

John Starkenberg
Kelly J. Benjamin

ARL-TR-645

December 1994

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 1994	3. REPORT TYPE AND DATES COVERED Final, 1 October 1992–30 June 1994

4. TITLE AND SUBTITLE Predicting Coalescence of Blast Waves From Sequentially Exploding Ammuntion Stacks	5. FUNDING NUMBERS 4G031-402-T5
6. AUTHOR(S) John Starkenberg and Kelly J. Benjamin	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: AMSRL-WT-TB Aberdeen Proving Ground, MD 21005-5066	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: AMSRL-OP-AP-L Aberdeen Proving Ground, MD 21005-5066	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARL-TR-645
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The current requirement that quantity-distance computations for air blast limitations be based on the total weight of all mass-detonating explosives at a storage site may be too restrictive. Therefore, we developed a computer program called "BWACO," intended to estimate pertinent aspects of the blast environment associated with sequentially detonating, spatially distributed ammunition stacks. This report explains the assumptions used and documents the evolution of BWACO on the Cray following its initial implementation. Comparison of preliminary results with experimental data obtained by Zaker led to replacement of the standard initially used for the description of blast waves with a new standard based on experimental data. BWACO has been adapted for the personal computer with enhanced graphical representations. Application to a number of problems representative of typical ammunition storage configurations are detailed. The results indicate that regions of significant pressure associated with the coalescence of blast waves from distributed ammunition stacks may be less extensive than corresponding regions associated with regulatory requirements. An advantage associated with the distribution of ammunition into smaller subdivisions was also demonstrated. As currently configured, BWACO provides a means of assessing the blast environment associated with the sequential detonation of an arbitrary arrangement of ammunition stacks. The limitations imposed by the assumptions have not been assessed in realistic configurations.

14. SUBJECT TERMS blast, blast waves, blast loads	15. NUMBER OF PAGES 108
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

INTENTIONALLY LEFT BLANK.

ACKNOWLEDGMENT

The work reported herein was supported under the Safeload Program by the Project Manager for Ammunition Logistics, Picatinny Arsenal, NJ. Safeload is ably managed by Mr. Robert A. Rossi.

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

INTENTIONALLY LEFT BLANK.

LIST OF FIGURES

Figure Page

INTENTIONALLY LEFT BLANK.

LIST OF TABLES

INTENTIONALLY LEFT BLANK.

1. BACKGROUND

In most instances, Army Regulation (AR) 385-64 requires that quantity-distance computations to determine ammunition storage siting limitations be based on the total weight of all mass-detonating explosives at the site. This requirement is made under the assumption that the blast waves emanating from storage subdivisions (ammunition "stacks") as they sequentially detonate will coalesce with one another before their amplitudes become negligible. If it can be shown that barriers between stacks prevent or sufficiently delay propagation of detonation between them, distances may be determined using the explosive weight of each stack individually. The regulation provides a crude criterion, based on initiation delay, for determining when coalescence will occur for the explosion of two closely spaced stacks. This criterion has not been verified for complicated ammunition storage arrangements and may be too restrictive, particularly where significant spatial separation of stacks exists.

Experience with the quantity-distance requirements has shown that they are often difficult to meet and waivers of the regulations have been frequently requested. If the actual coalescence pattern from a complex array of ammunition stacks could be estimated, a more accurate assessment of the hazards produced could be made, and significant reductions in real-estate requirements might result. In principle, this could be accomplished by numerical simulation of the blast environment in the region of interest. However, this approach requires immense computer resources, even for the simplest arrangements, and is beyond the reach of the typical user. Alternatively, approaches based on simplified analyses which have been developed in the nuclear blast community could be applied. Specifically, algorithms found in the DNA Nuclear Blast Standard (1 KT) (Needham and Crepeau 1981), a computer program which describes the blast environment produced by a 1-KT nuclear explosion, and the Low Altitude Multiple Burst (LAMB) model, a computer program which describes the blast environment produced by multiple nuclear explosions of arbitrary yield, served as the starting point for the present approach.

2. APPROACH

We originally intended to adapt these programs to our present purposes. However, direct use of LAMB required the performance of many unnecessary, time-consuming computations. Therefore, we developed a new application called BWACO, intended to estimate pertinent aspects of the blast environment associated with sequentially detonating, spatially distributed ammunition stacks.

In its initial implementation, BWACO made significant use of 1-KT Standard and LAMB algorithms. However, as we investigated the performance of the code, these were gradually abandoned in favor of new algorithms. The main development effort was conducted using ARL's Cray X/MP 48 because graphics software was readily available. However, once a FORTRAN-callable graphics package was obtained, the code was converted for use on IBM Personal Computers (PCs) or compatible machines.

This report explains the assumptions used and documents the evolution of BWACO on the Cray following its initial implementation. Features added in the PC version are illustrated through application to a number of problems representative of typical ammunition storage configurations. A detailed description of the code (including subroutine listings) and the initial versions of the BWACO User's Manual are presented in the appendices.

3. BWACO ASSUMPTIONS

Sequential detonations are assumed to occur as a result of propagation of detonation from one stack to another. The first stack of any such pair to detonate is called the donor and the second is called the acceptor. The first of all the stacks to detonate is referred to as the initial donor. Detonation propagation is assumed to occur at a constant velocity which is a property of the donor. This is consistent with a fragment impact mechanism at least over distances short enough to preclude significant deceleration of the fragments. The velocity provided must be that of the fastest fragment produced by munitions in the stack augmented to account for focussing effects. The fact that no consideration has been given to aerodynamic fragment decelerations or to additional delays associated with the penetration of barriers makes this a worst-case assumption (when sufficiently high fragment velocities have been specified).

Classical blast wave structure is assumed. The blast wave consists of a shock wave and positive overpressure phase followed by a negative overpressure phase (when sufficiently far from the source). The usual scaling laws are assumed to apply and are used to determine required blast wave characteristics with reference to a standard. BWACO requires shock and zero-overpressure arrival times at specified stations. The 1-KT Standard gives the shock and zero-overpressure radii as functions of time based on fits to analytical solutions. These functions may be inverted using Newton's method to obtain the arrival times. We developed alternative fits to experimental data for a somewhat different standard which provides the shock arrival time and positive phase duration as functions of position. These can be used directly to compute shock and zero-overpressure arrival times.

The ground plane is assumed to be perfectly flat and rigid and to effectively double the explosive weight of a stack. The latter is also a worst-case assumption. Waves from multiple sources are assumed to propagate independently from one another, and coalescence is assumed to occur wherever and whenever the shock associated with one blast wave encroaches into the positive overpressure phase associated with another blast wave.

Under these assumptions, regardless of the spatial and temporal separation between the sources, at sufficiently large distances from the charges, coalescence will be predicted. Blast waves from actual detonating stack arrangements do not exhibit this property because they do not propagate independently. Rather, portions of some of them propagate in air processed by their predecessors. An overtaking wave may first encounter the negative overpressure phase of the wave being overtaken causing it to decelerate. The overtaking wave must be somewhat stronger than the wave being overtaken or it will not be able to penetrate the negative overpressure phase and coalescence cannot occur. Once this phase has been penetrated and the overtaking wave has entered the positive phase of the wave being overtaken, coalescence is assured but occurs only after some additional propagation. For these reasons, the BWACO coalescence algorithm can be expected to predict coalescence where, in actuality, it does not occur. A margin of safety is provided by prediction of a larger hazard area than actually exists.

We considered several methods of combining peak overpressures from N coalesced waves. These include simple superposition,

$$\Delta p = \sum_{i=1}^{N} \Delta p_i ,$$

the full LAMB algorithm,

$$\rho = \rho_o + \sum_{i=1}^{N} \Delta\rho_i ,$$

$$\overline{V} = \frac{\sum_{i=1}^{N} \rho_i \overline{V}_i}{\rho} ,$$

$$\Delta p = \sum_{i=1}^{N} \Delta p_i + \frac{\gamma+1}{2}\left(\frac{1}{2}\sum_{i=1}^{N} \rho_i V_i^2 - \frac{1}{2}\rho V^2\right),$$

or use of the pressure produced by the total explosive weight of all stacks associated with the coalesced waves combined at their center of charge. This last approach is described in greater detail in section 5.5.

4. USING BWACO

In order to run BWACO, the user supplies a description of the distributed ammunition stacks and, if desired, of the region of interest. The program divides the region into discrete stations. It is recognized that, generally, any stack may act as the initial donor and it is necessary to determine the worst-case loading (highest combined peak overpressure) at each station considering all initial donors. For each initial donor, BWACO determines the timing of detonation of each of the other stacks. It then applies the coalescence criterion and computes both the total equivalent weight of the explosive in the stacks contributing to the blast and the peak overpressure at each station. It retains the values associated with the highest peak overpressure for any initial donor. The results are reported in graphical form.

User specifications are supplied by means of an appropriately formatted input file. Each ammunition stack is defined by specifying its coordinates in a user-defined Cartesian system, its net equivalent explosive weight, a propagation velocity for communication of mass detonation to neighboring stacks, and whether or not the stack may act as an initial donor. Any convenient Cartesian coordinate system may be used with distances specified in meters. The explosive weight for the TNT equivalent, in pounds, of all the explosive in the stack is required, as is the propagation velocity in m/s. Usually, in order to allow determination of the worst-case blast loading, all of the stacks should be allowed to act as initial donors. However, this input option provides the flexibility required to address special problems such as simulating experiments with known donors. The extent of the region of interest is specified using the same coordinate system defined for the stacks. If no region is specified, BWACO selects a region limited to the area in which the total equivalent weight of the explosive in all the stacks produces peak blast overpressures above the allowable value for inhabited buildings (unless similar considerations for the individual stacks dictate a larger region). The program then positions 3,600 stations within the region.

BWACO cycles through all possible initial donors as specified by the user. For each initial donor, using the given propagation velocities, it computes the times at which the other stacks detonate. It applies the coalescence criterion and, if necessary, combines the peak overpressures at each station.

BWACO provides different graphical representations of the results depending on the version used. The coalescence map is a plot showing areas of coalescence within the region of interest and indicating, in the Cray version, the number of waves which have coalesced or, in the PC version, the total equivalent explosive weight contributing to the blast. The peak overpressure map, available only in the PC version, is a contour plot showing the worst-case peak overpressure from the blast in the region of interest. The contour intervals correspond to the five permissible exposure levels defined in AR 385-64.

5. EVOLUTION OF THE BWACO MODEL

5. 1 Review of Experimental Results. Zaker (1969) reported results of an extensive analytical and experimental study of the coalescence of blast waves produced by pairs of sequentially detonating Composition C-4 charges having a total weight of two pounds. In the analysis, the charges were assumed to be located at the same point, while in the experiments they were separated by 10 inches and a steel barrier to prevent sympathetic detonation. Pressure was measured along a lateral line equidistant from the charges and along an axial line. The analysis was assumed to be applicable to the experiments along the lateral line. The axial values measured depend on the separation between the charges and are affected by the presence of the barrier, while the lateral values are approximately independent of the separation and are predicted by the analysis.

Zaker considered charge weight ratios for successively detonated charges of 1:2, 1:1, and 2:1 and nominal initiation delay times ranging from 0.8 to 5.7 ms corresponding to 0.60 to 4.11 ms/lb$^{1/3}$ using conventional blast scaling. From this it can be inferred that an equivalent TNT weight of about 2.62 lb was used in the scaling. The blast environment was monitored using pressure gauges on the axis of charge centers and in the lateral direction perpendicular to this axis at the center of charge to a distance of approximately 58 ft (42 ft/lb$^{1/3}$). This distance is a little shorter than that at which the overpressure from the combined charges decays to about 0.9 psi. This represents the smallest permissible exposure level (which applies to inhabited buildings) defined in AR 385-64.

Pressure records showing the coalescence process were presented by Zaker. These indicate that substantial propagation, covering most of the field of observation, may occur as coalescence progresses. He observed that the peak overpressures associated with coalesced waves were essentially the same as those produced by single charges of the same total explosive weight.

For two charges of equal mass, Zaker found that a "tendency" to coalescence in the lateral direction occurs for delays of less than 4.3 ms (3.2 ms/lb$^{1/3}$). The term "tendency" was not clearly defined but it may be assumed to mean observation of decreasing intervals between wave peaks with increasing distance from the charge center. This is actually very similar to the BWACO criterion. He observed a tendency to coalescence in the axial direction with all of the delays considered (up to 5.7 ms or 4.1 ms/lb$^{1/3}$). By comparing lateral and axial data, he estimated that the effect of the charge separation and the barrier on the interval between peaks in the axial direction is equivalent to an additional delay of 1.8 ms (1.3 ms/lb$^{1/3}$). Thus, axial coalescence may be assumed to occur for delays less than 6.1 ms (4.5 ms/lb$^{1/3}$). This result may include significant effects of the barrier.

For charges of unequal mass, Zaker observed that coalescence persists at longer delays with a weight ratio of 1:2 and vanishes at shorter delays with a weight ratio of 2:1 compared to equal charges. (Thus, the criterion given in AR 385-64 uses more conservative values of 4.0 and 5.6 ms/lb$^{1/3}$, respectively.) Zaker also performed some experiments with three equal charges. He observed that the third pulse tends to overtake the second before the second overtakes the first.

5.2 Simulation of the Experiments Using the 1-KT Standard. In order to simulate Zaker's equal-charge experiments, we made BWACO computations for two 1.31-lb "stacks" separated by 0.254 m, specifying only one possible initial donor. Specific delay times were obtained using appropriate values of the propagation velocity. The region of interest was specified as extending to 18.0 m from the center of charge in each direction, corresponding to the region of experimental observation. The delay times associated with the vanishing of coalescence at the edges of the region may be compared with experimental values.

Coalescence maps of regions of the surface for various delay times are plotted in Figure 1. The donor (solid circle) and acceptor (solid square) are plotted near the center of the map and overlap. Regions of coalescence are shaded with dots. Unshaded areas experience two independent waves, while shaded areas experience a single coalesced wave. For delays less than 3.3 ms, coalescence within the region of interest is predicted in all directions. For delays greater than 4.0 ms coalescence in the lateral (x) direction vanishes. Thus, with a delay of 4.3 ms (Zaker's limit for lateral coalescence), coalescence in the lateral direction is not predicted. Coalescence in the axial (y) direction persists until the delay exceeds 4.7 ms. This may be compared to Zaker's value of 6.1 ms. The difference between the delays for which coalescence vanishes in the axial and lateral directions respectively is only 0.7 ms, which is much less than Zaker's value of 1.8 ms.

In each direction, vanishing of coalescence is predicted at shorter delays than observed in or estimated from the Zaker experiments. The discrepancy is most significant in the axial direction. This result, which

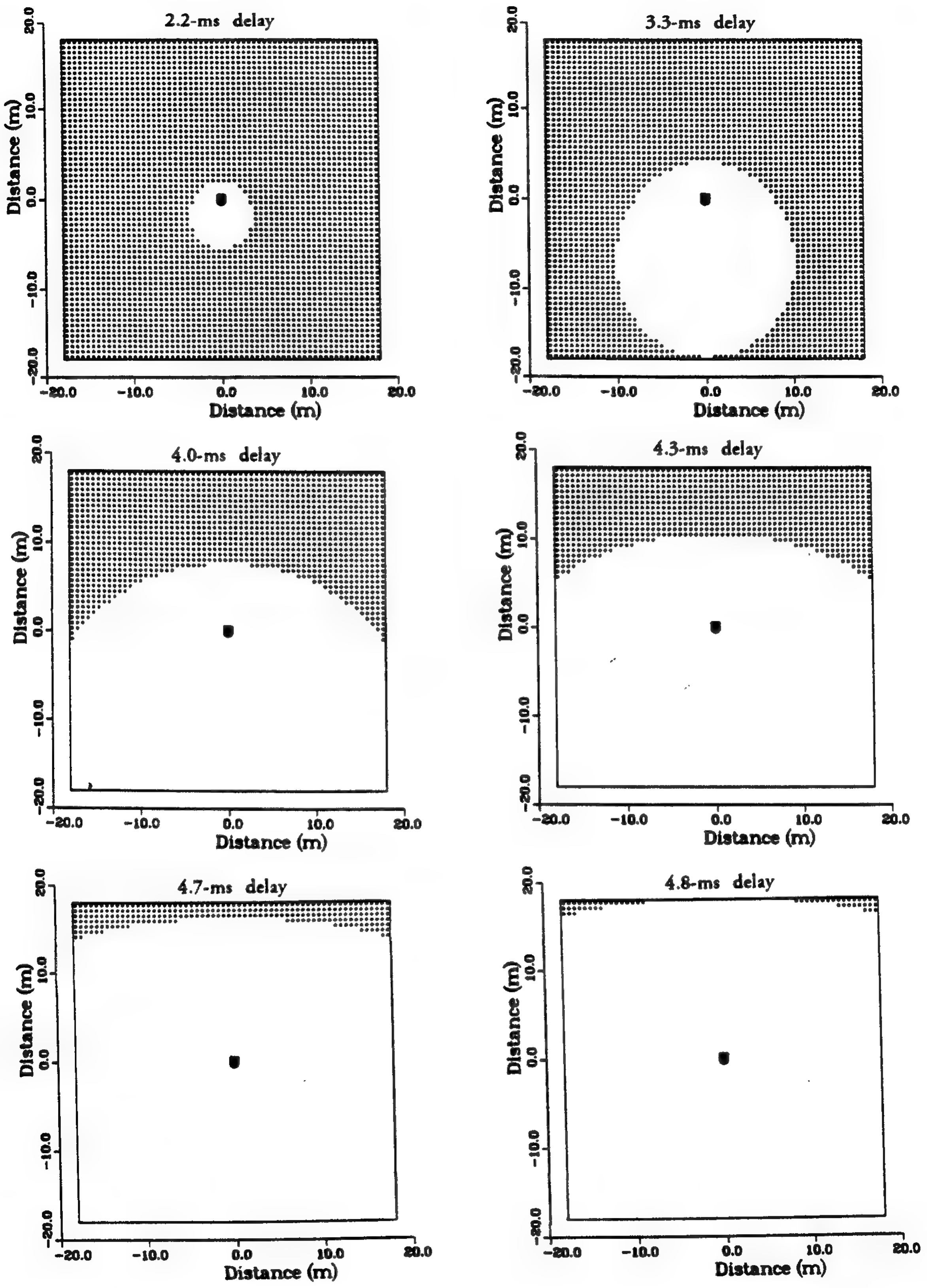


Figure 1. Coalescence maps for Zaker experiment simulations using the 1-KT standard.

is unacceptable, contradicts our expectations for the bias in BWACO's coalescence algorithm and suggested to us that the 1-KT Standard might be inaccurate.

5.3 Alternatives to the 1-KT Standard Fits. Kingery and coworkers have reported air blast parameters, including shock arrival time, positive phase duration and peak overpressure, versus distance for large-scale hemispherical TNT surface bursts (Kingery, Keefer, and Day 1962; Kingery and Pannill 1964; and Kingery 1966). This data provides a basis for evaluating the applicability of the 1-KT Standard. Comparison with the 1-KT Standard shock arrival time is shown in Figure 2. There is close agreement except in a tiny region very near the source. Comparison with the 1-KT Standard positive phase duration is shown in Figure 3. The values from the standard are generally too short at distances of interest. This would tend to suppress the prediction of coalescence, producing the results we observed. Comparison with the 1-KT Standard peak overpressure is shown in Figure 4. Zaker's small-scale results are also plotted. The 1-KT Standard pressures are generally lower than the measured values.

Alternative fits which give shock arrival time and positive phase duration as functions of distance were developed. These retain the appropriate asymptotic behavior at large distances from the source. The fit for shock arrival time is of the form

$$t_a = a_a[r - b_a \tanh(\frac{r}{r_a})],$$

where a_a, b_a and r_a are constants. The fit for positive phase duration is of the form

$$\Delta t_+ = a_+(1 - \frac{b_+}{r^{c_+}})[\ln(r^2 + r_+^2)]^{1/2},$$

where a_+, b_+, c_+ and r_+ are constants. This fit is not used for very small values of r, for which the value of the positive phase duration is held constant. Kingery and Pannill (1964) provided a fit for peak overpressure. These fits are also plotted in Figures 2 through 4 and provide improved agreement with measurements in each case.†

†After implementation of this standard, we learned of the existence of a TNT standard.

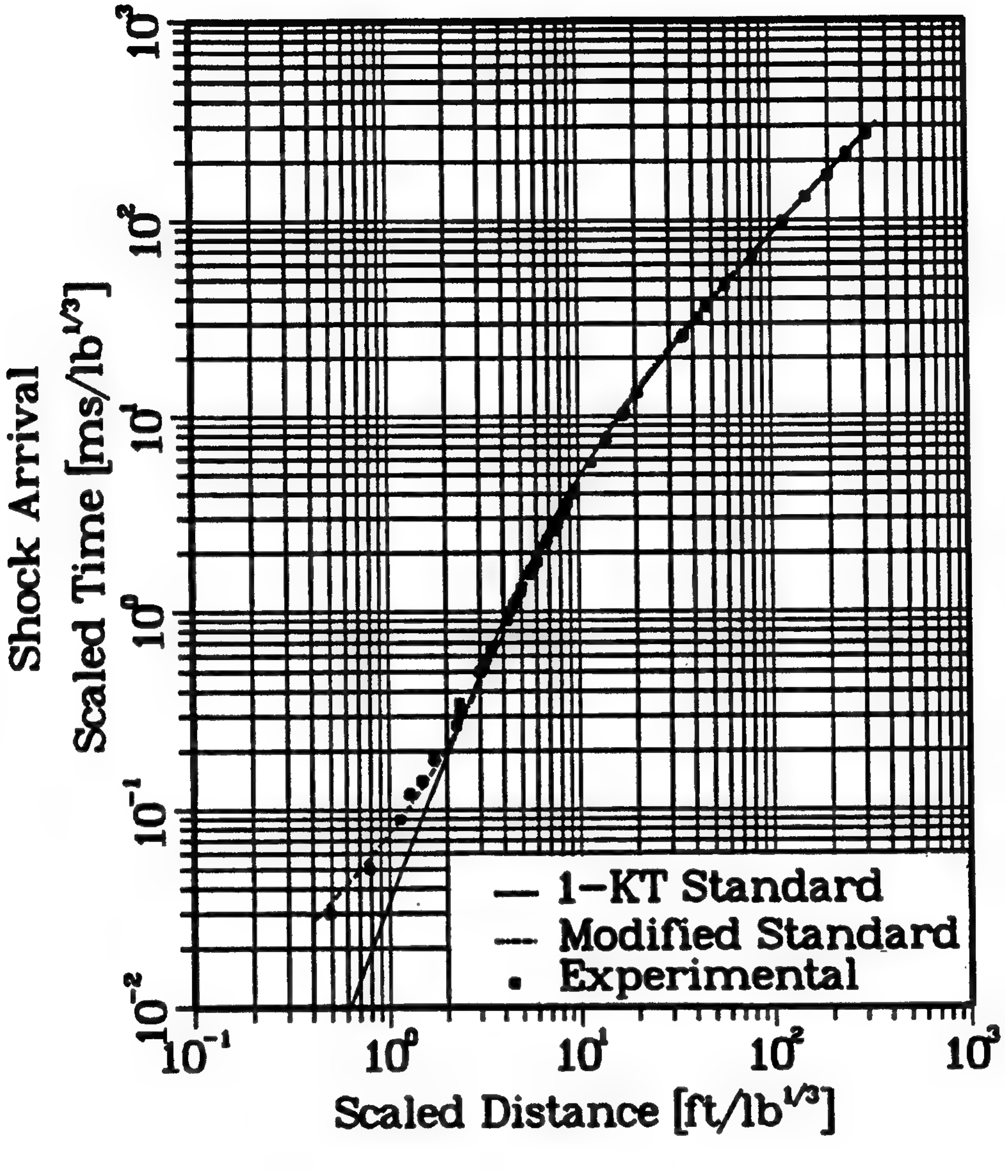


Figure 2. Comparison of the 1-KT standard with the alternative standard for shock arrival time.

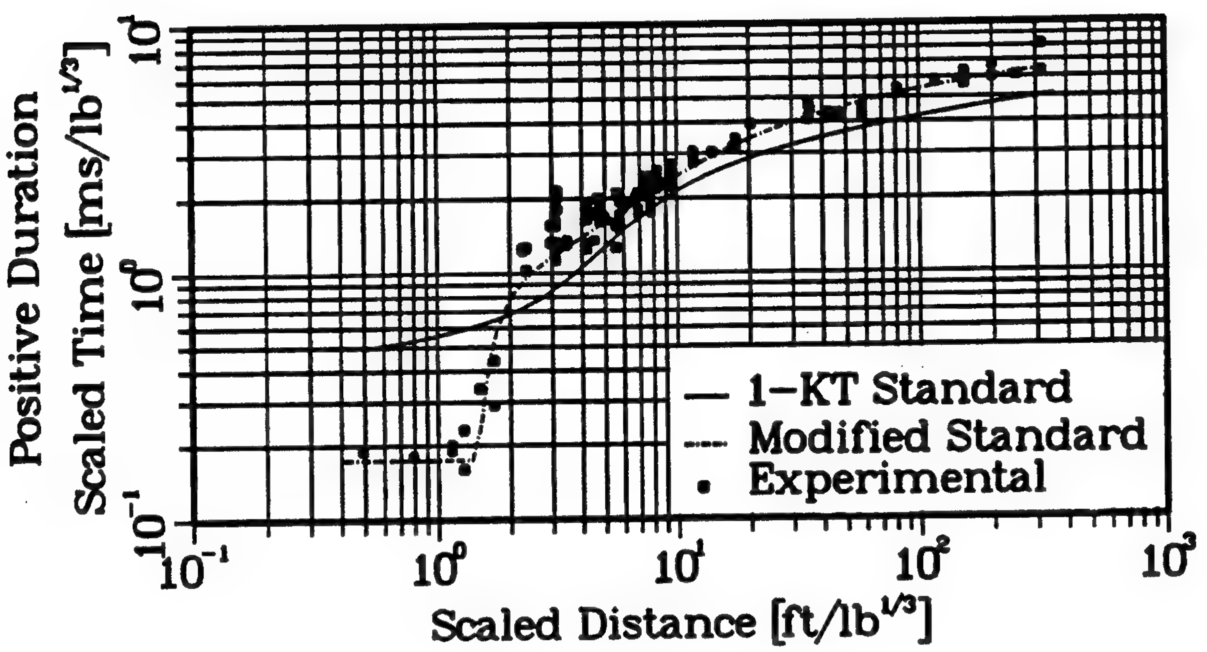


Figure 3. Comparison of the 1-KT standard with the alternative standard for positive phase duration.

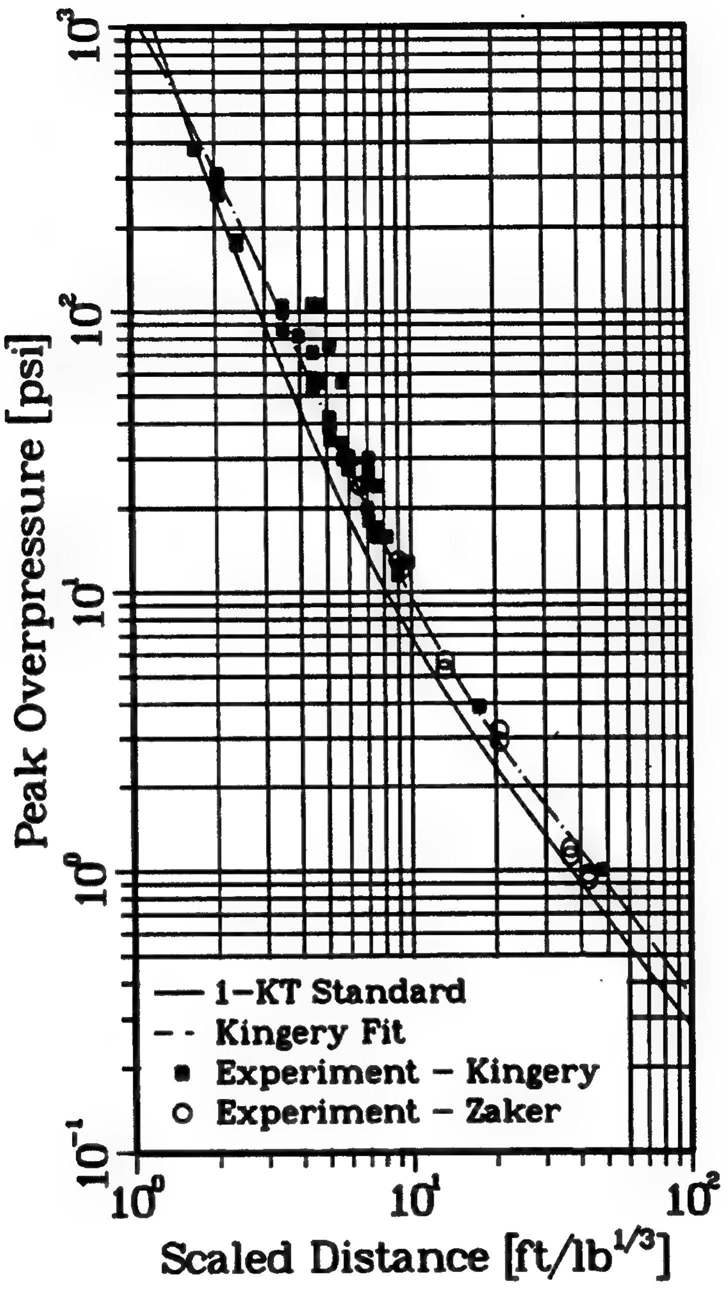


Figure 4. Comparison of the 1-KT standard and the Kingery fit with the Kingery and Zaker data for peak overpressure.

5.4 Simulation of the Experiments Using the Alternative Standard. Coalescence maps obtained using the alternative standard are shown in Figure 5. Lateral coalescence vanishes for delays exceeding about 5.1 ms, which is significantly greater than Zaker's observation of 4.3 ms. In the axial direction, coalescence vanishes for delays of 5.9 ms or greater. This is only slightly less than Zaker's estimate of 6.1 ms. There is no significant change in the difference between the axial and radial values. It is still less than half that determined by Zaker. The predictions obtained with the alternative standard do not miss significant regions of coalescence and are acceptable.

As we have noted, at sufficiently large distances from the stacks, coalescence will be predicted. For the present conditions, this becomes apparent upon expanding the regions of interest while maintaining the other parameters as shown in Figure 6. Although coalescence is identified at large distances, the pressures produced are too low to be of concern. BWACO was configured to suppress identification of coalescence where the combined peak overpressure is below 0.9 psi (the maximum permissible exposure level for inhabited buildings subjected to blast waves from large charges). The effect on the coalescence maps is to eliminate shading where distance from the charges is sufficiently large as illustrated in Figure 7. It appears that the pressures combined using simple superposition are somewhat high, as the extent of the region of significant pressure is greater than that associated with a single charge of the same total weight. As a result of the expanded extent, coalescence is observed at somewhat longer delays than those associated with the experimental region of significant pressure specified in the earlier computations. (See Figure 5.)

5.5 Effect of Pressure Combination Algorithms. To assess the effect of the pressure combination algorithms on the extent of the region of significant pressure, we made computations with from 2 to 10 stacks, located at the same point, having the same total equivalent explosive weight (2.62 lb). The ratio of the radius of the region of significant pressure produced by the detonation of n stacks to that produced by the detonation of one stack is plotted as a function of n in Figure 8. Contrary to Zaker's observation, using simple superposition, we obtain substantial increase in the extent of the region of significant pressure with increasing number of stacks even though the total explosive weight remains constant.

The results obtained with the LAMB algorithm (also shown in Figure 8) are virtually identical to those obtained using simple superposition. The LAMB algorithm has been shown to produce correct results for the head-on collision of two equal blast waves and overestimates the pressure in other cases (Hikida and

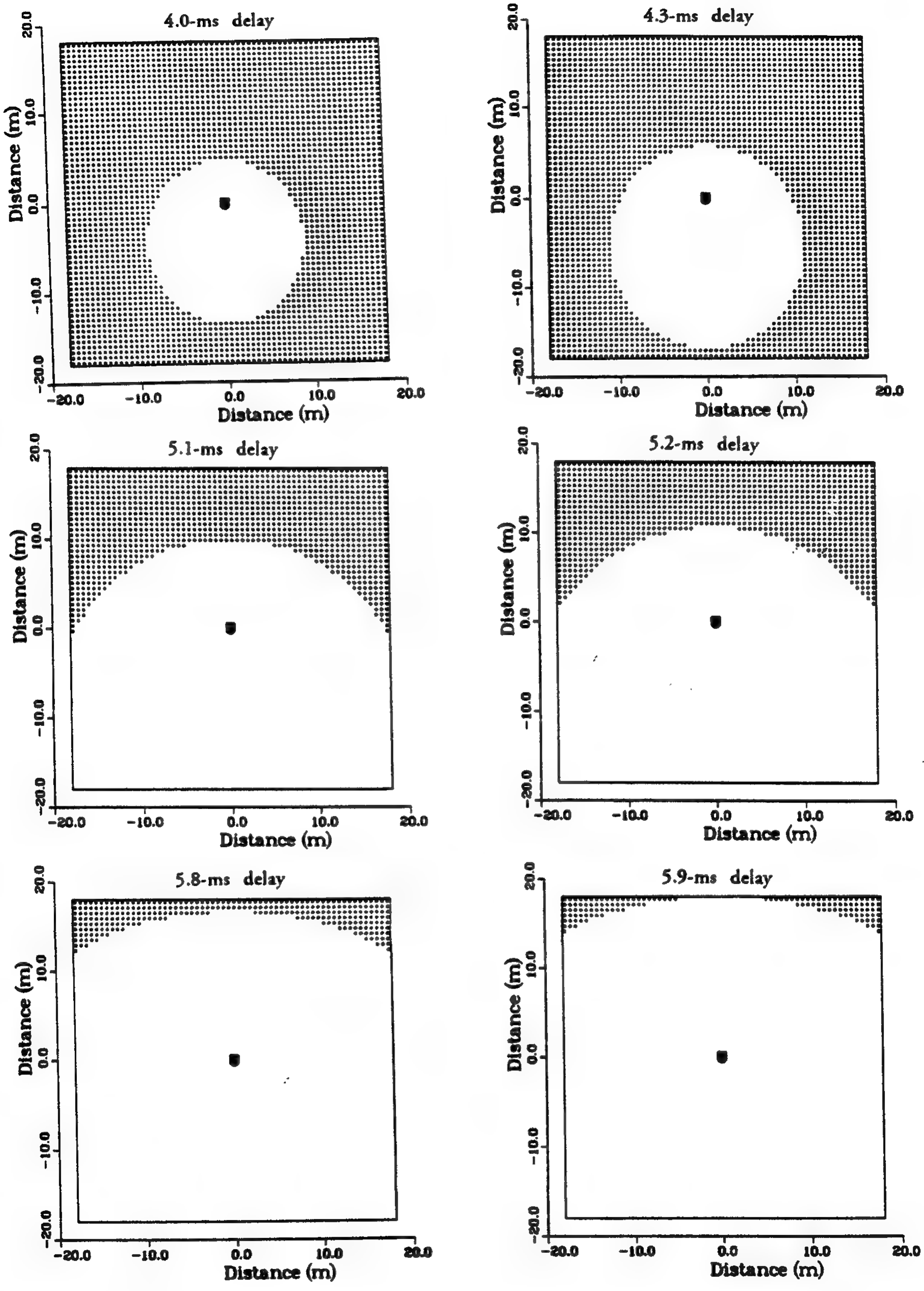
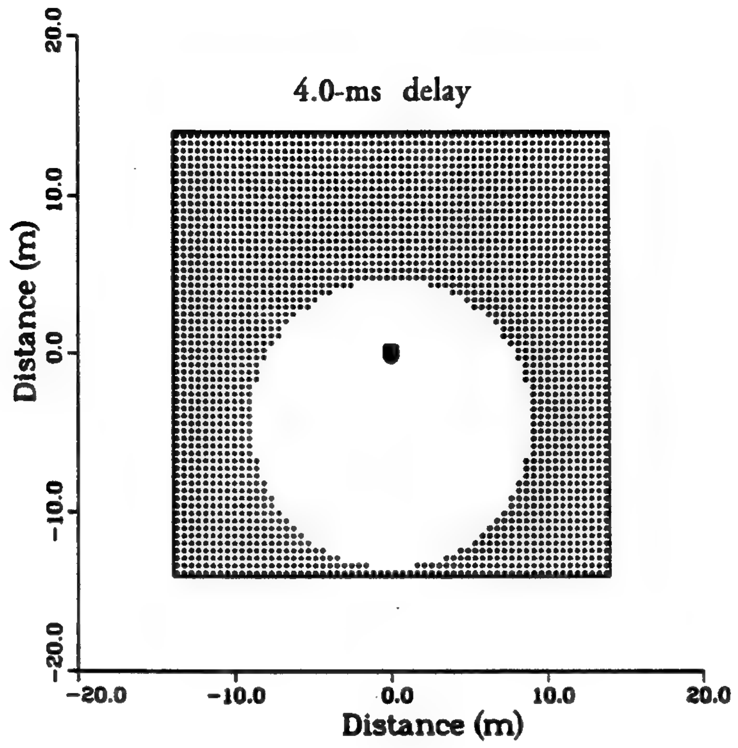


Figure 5. Coalescence maps for Zaker experiment simulations using the alternative standard.



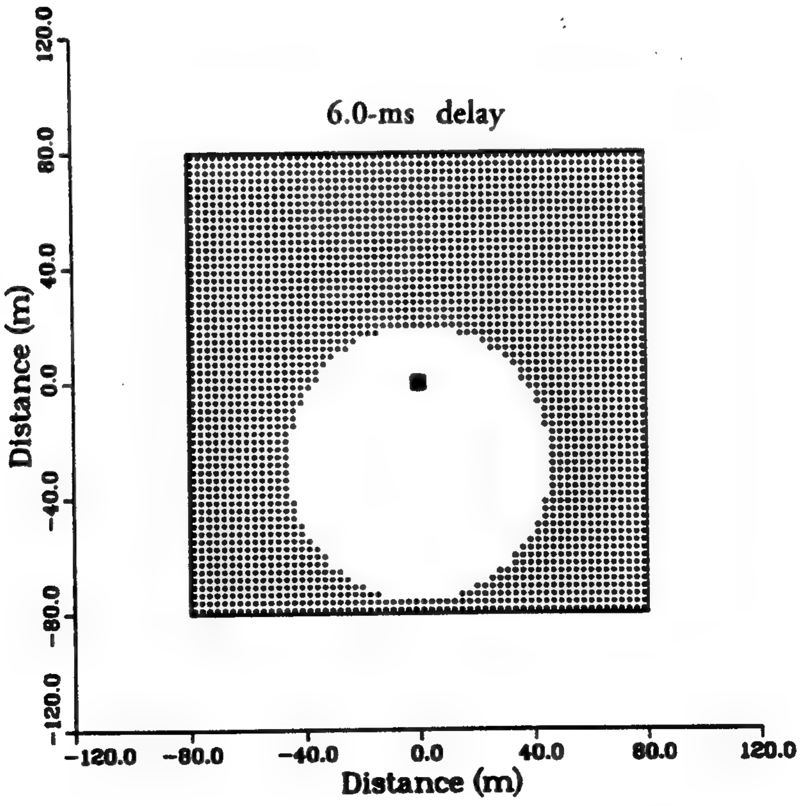


Figure 6. Coalescence maps showing the effect of scale on coalescence at two different delay times.

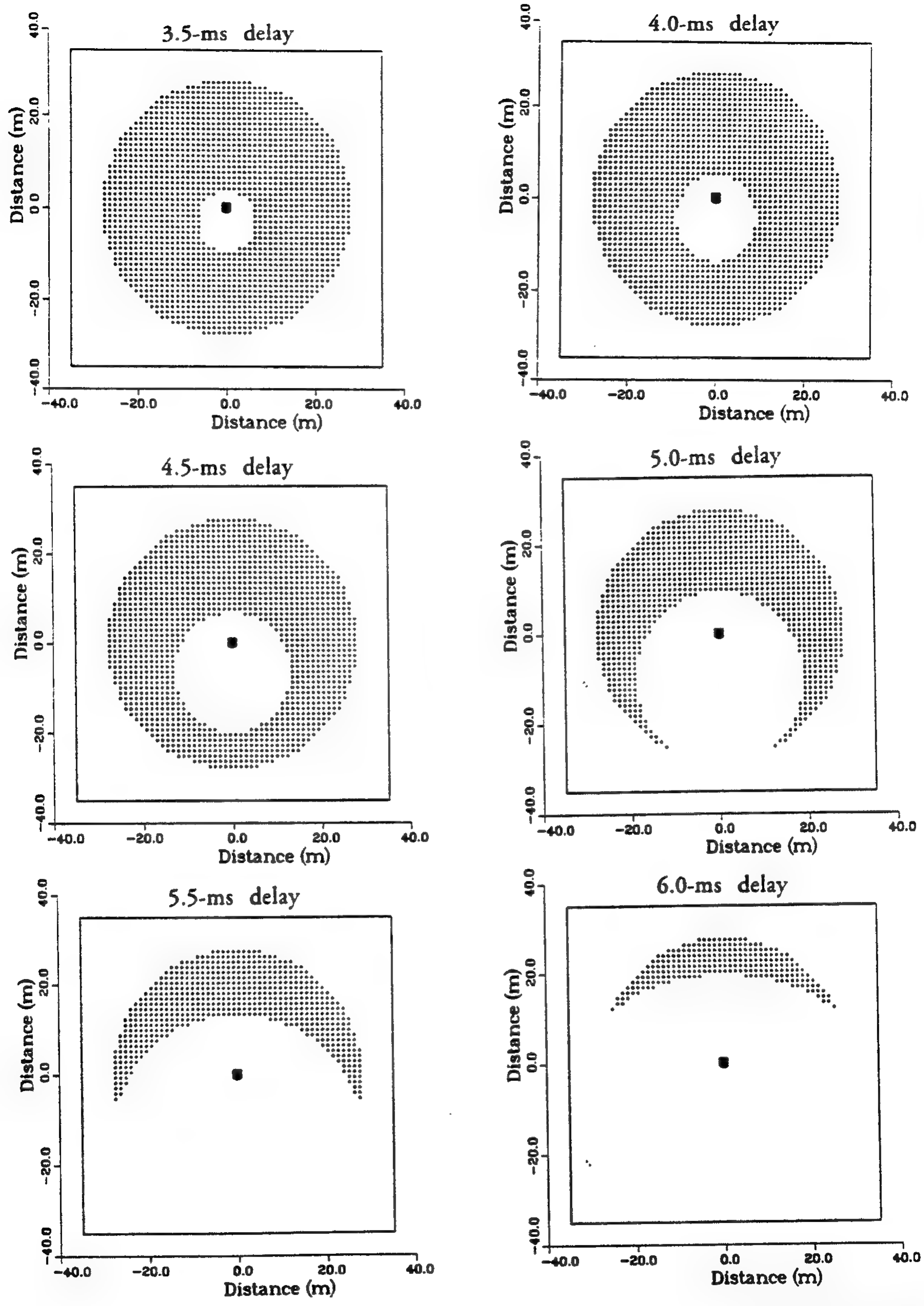


Figure 7. Coalescence maps for Zaker experiment simulations with coalescence indications suppressed at low peak overpressure.

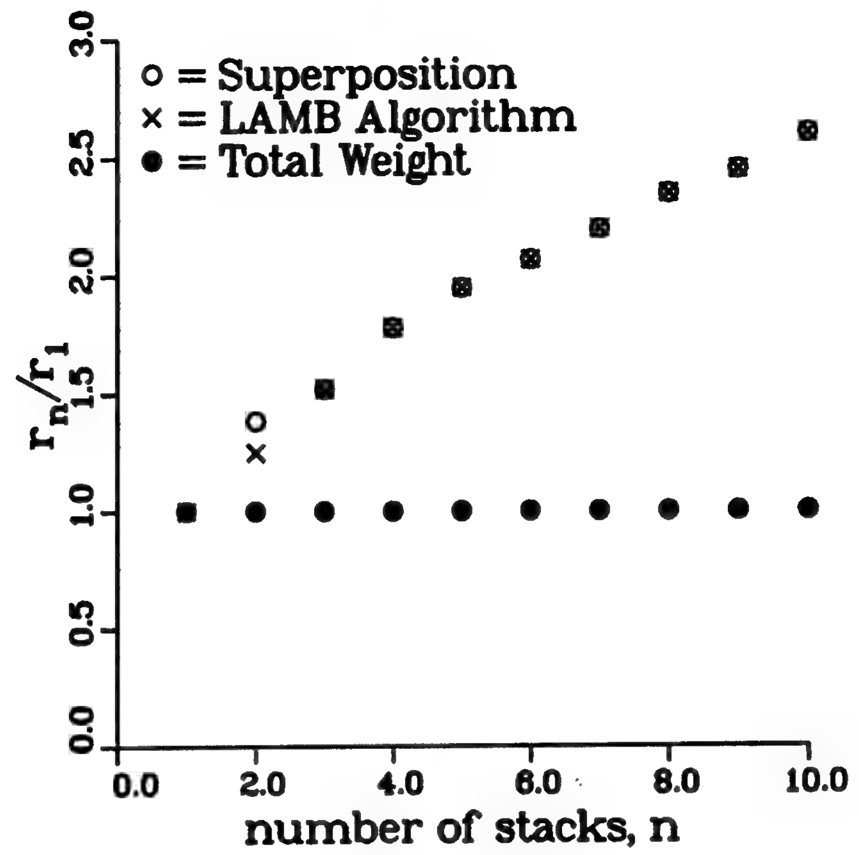


Figure 8. Comparison of the effects of the pressure combination algorithm on the region of significant pressure as a function of the number of charges.

Needham 1981; Brode 1977). In the case of overtaking waves, the magnitude of the overestimate appears significant.

As an alternative to the simple superposition and LAMB algorithms, we computed the combined pressure by first combining the stacks which produce the coalesced wave into a single stack positioned at the center of charge. This approach, suggested by Zaker's observation, is applicable where the angles between coalescing waves are small, as occurs at any point whose distance from the stacks significantly exceeds the distance between stacks. It produces regions of significant overpressure which do not increase with the number of stacks. This result is also shown in Figure 8.

5.6 Simulation of the Experiments Using the Alternative Standard and Pressure Combination Algorithm. The way in which the alternative pressure combination algorithm modifies the results obtained with two equal charges is shown in Figure 9. As expected, the region of significant pressure is reduced in comparison to that shown in Figure 7. The delays at which coalescence vanishes are slightly increased compared with those associated with the analysis shown in Figure 5 because the region of significant pressure is larger than the region of experimental observation.

We also made computations with unequal charges in ratios of 1:2 and 2:1 with a constant total equivalent weight of 2.62 lb. Figure 10 shows coalescence maps for the 1:2 case at delays ranging from 4.0 to 6.5 ms. There is still a small region of coalescence at the longest delay. Figure 11 shows coalescence maps for the 2:1 case at delays ranging from 3.0 to 5.5 ms. In this case, coalescence has completely vanished at the longest delay. Figure 12 compares results from each of these cases with the equal-charge case at two different delays. This shows that the persistence of coalescence decreases as the ratio of the donor weight to the acceptor weight increases. The result is consistent with Zaker's observations.

Figure 13 shows coalescence maps obtained in a simulation of Zaker's three-charge experiment. The size of each shading dot is proportional to the number of waves coalesced at the corresponding station. However, it cannot be ascertained which waves have coalesced. For short delays, coalescence of all three waves is predicted in all directions, although two-wave coalescence occurs first (closer to the center of charge). As the delay is increased, three-wave coalescence begins to vanish, starting in the negative axial direction. Two-wave coalescence persists for longer delays. The results are consistent with Zaker's observation that the third pulse tends to overtake the second before the second overtakes the first.

5.7 Results With Multiple Initial Donors. To this point, all the computations have specified a single donor. Each resultant coalescence map shows the coalescence pattern associated with a single chain of events. However, BWACO is generally intended to be used with all stacks specified as possible initial donors. The coalescence map produced in this case is a composite which reflects the worst case loading at each point in the region of interest. The effect of allowing each stack to act as an initial donor in the Zaker simulations was examined in order to check the code's multiple donor algorithm.

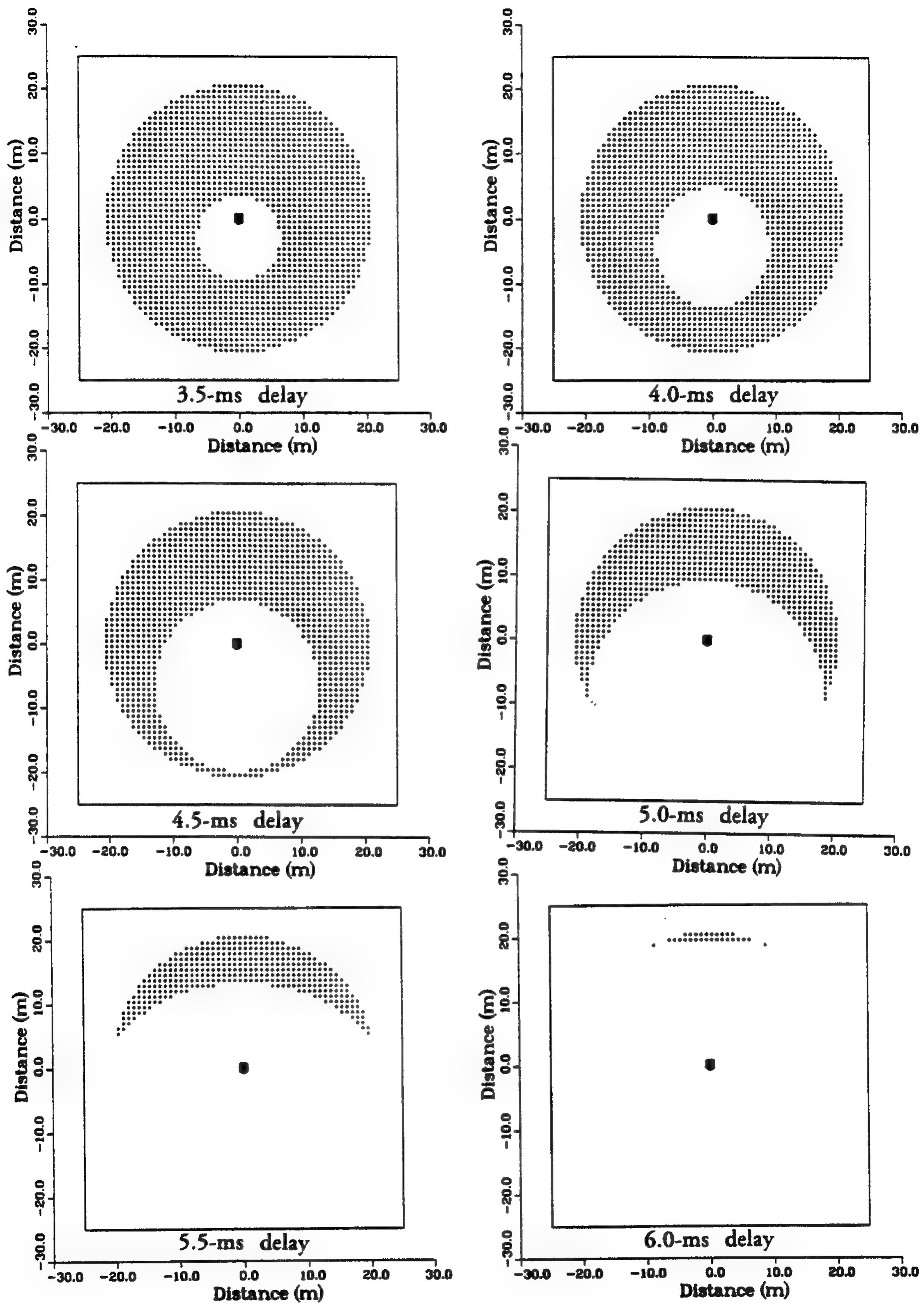


Figure 9. Coalescence maps for Zaker experiment simulations using the alternative standard and pressure combination algorithm.

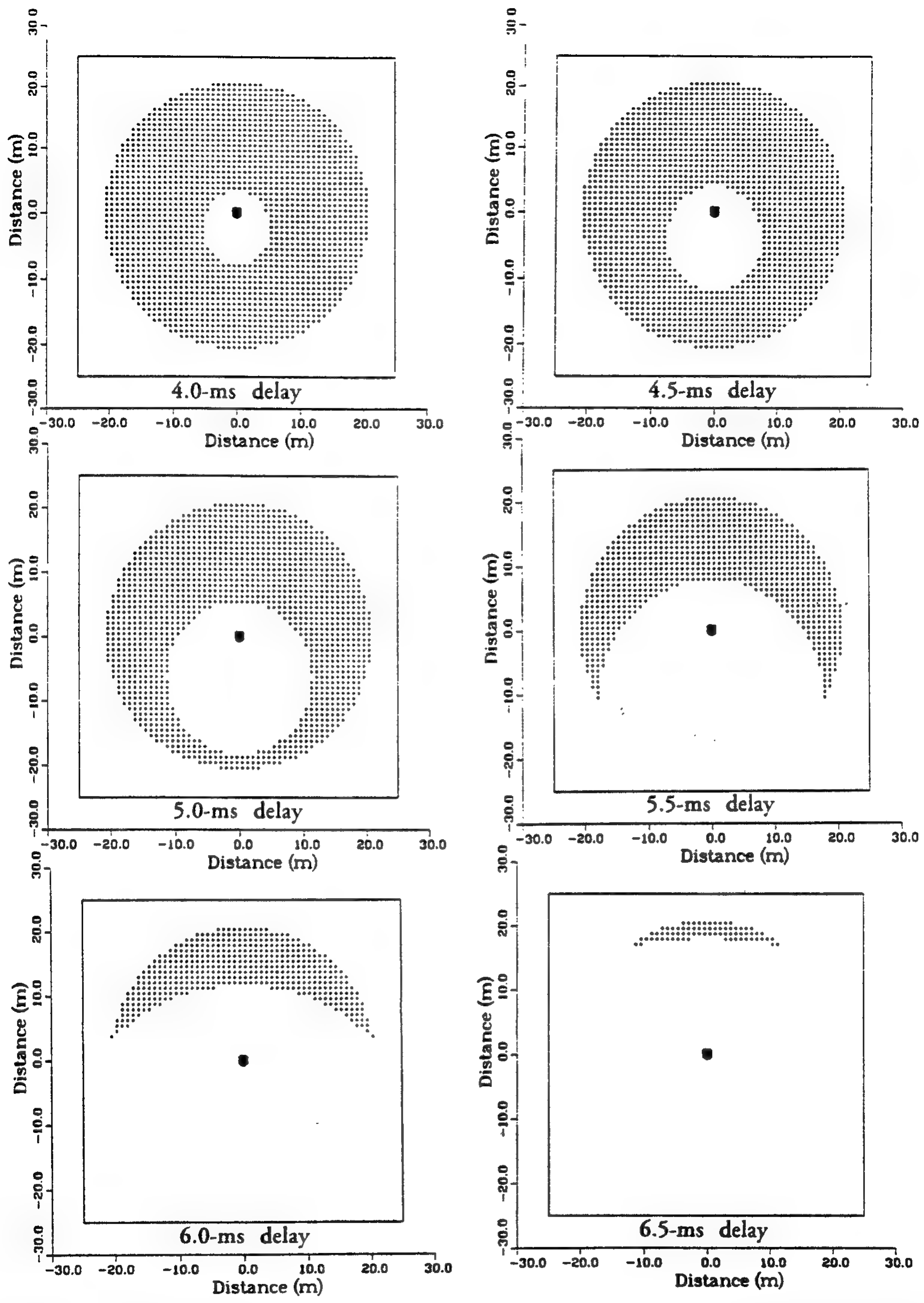


Figure 10. Coalescence maps for Zaker experiment simulations with unequal charges having weight ratios of 1:2.

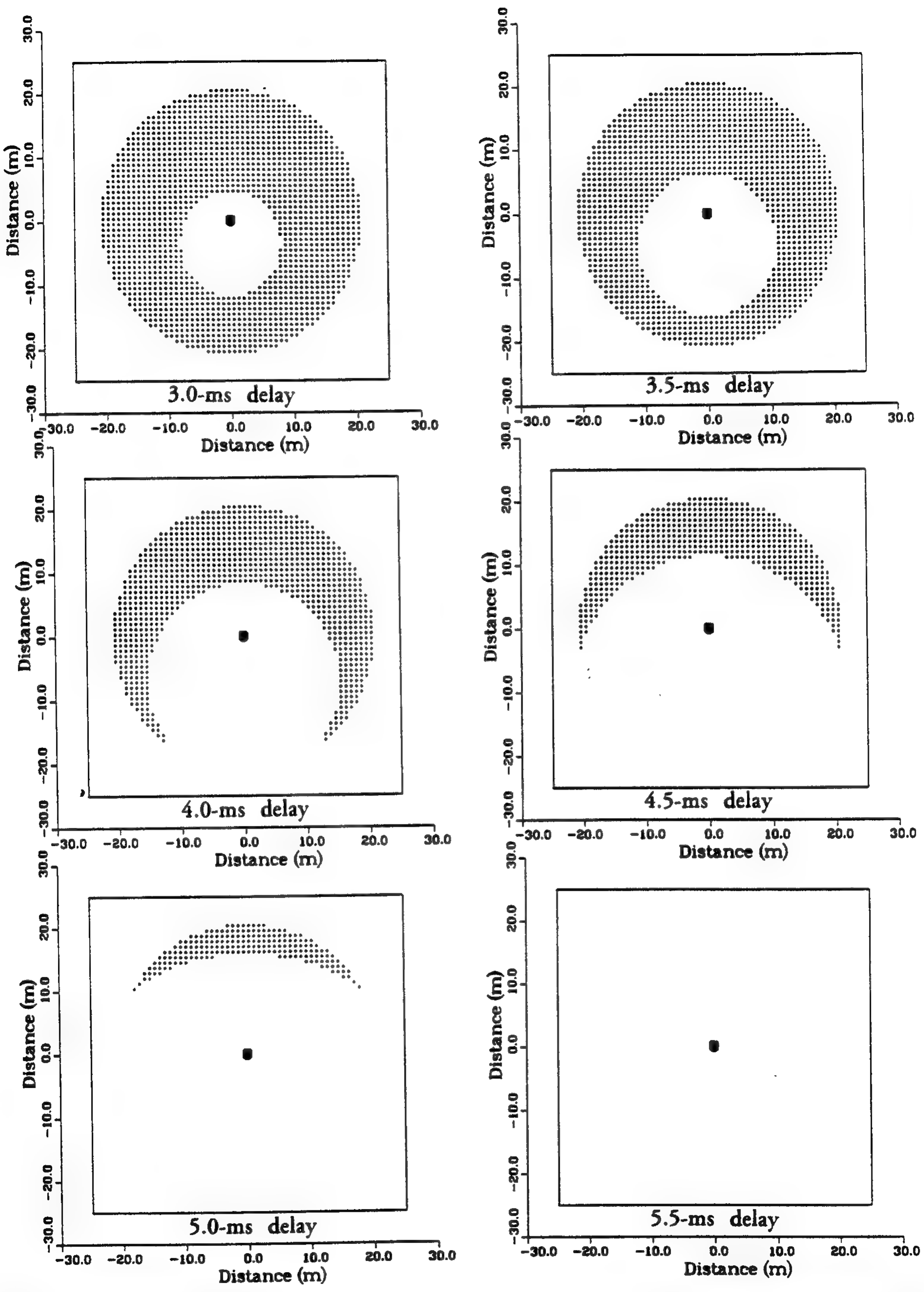


Figure 11. Coalescence maps for Zaker experiment simulations with unequal charges having weight ratios of 2:1.

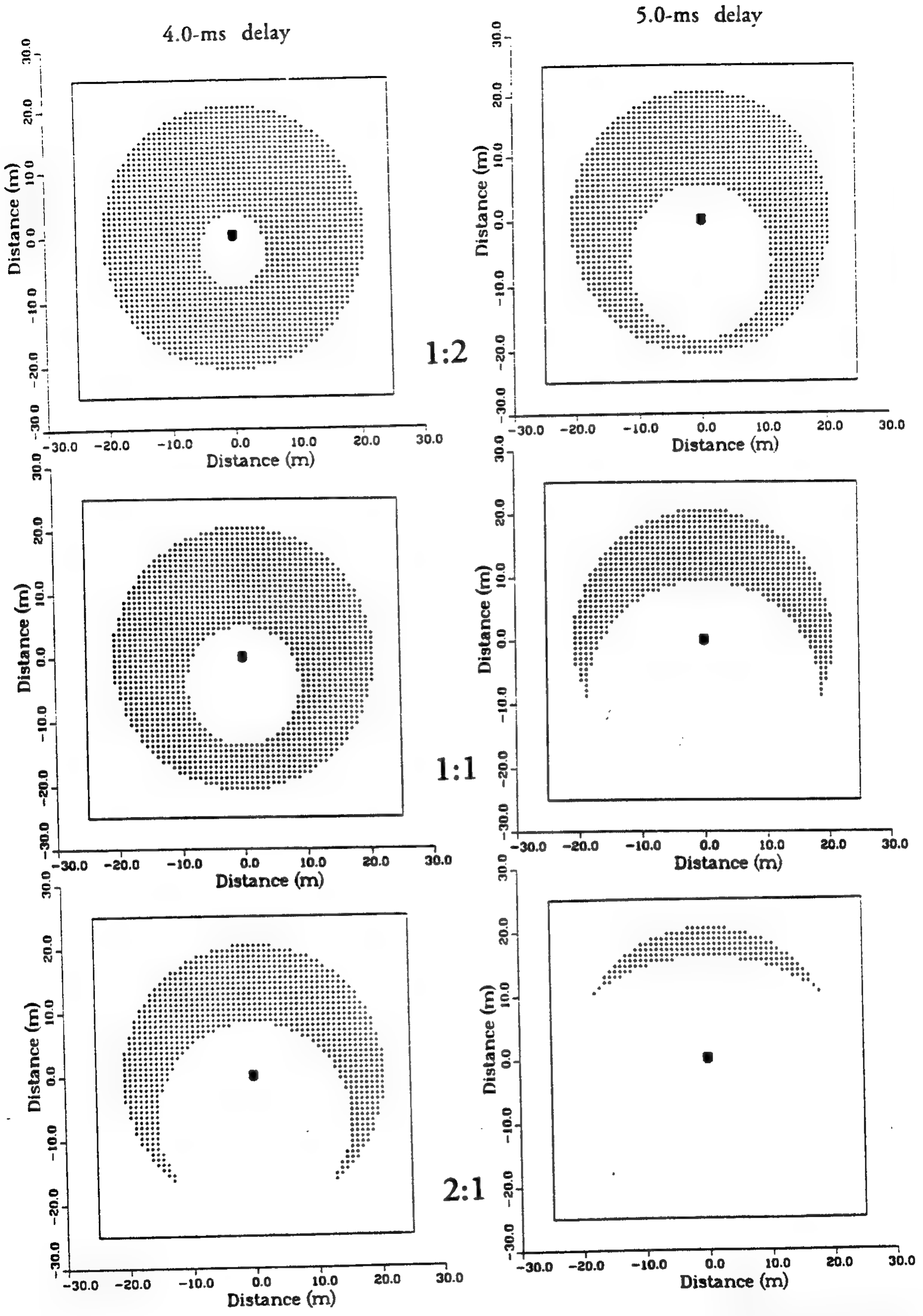


Figure 12. Coalescence maps for Zaker experiment simulations with unequal charges having weight ratios of 1:2 and 2:1.

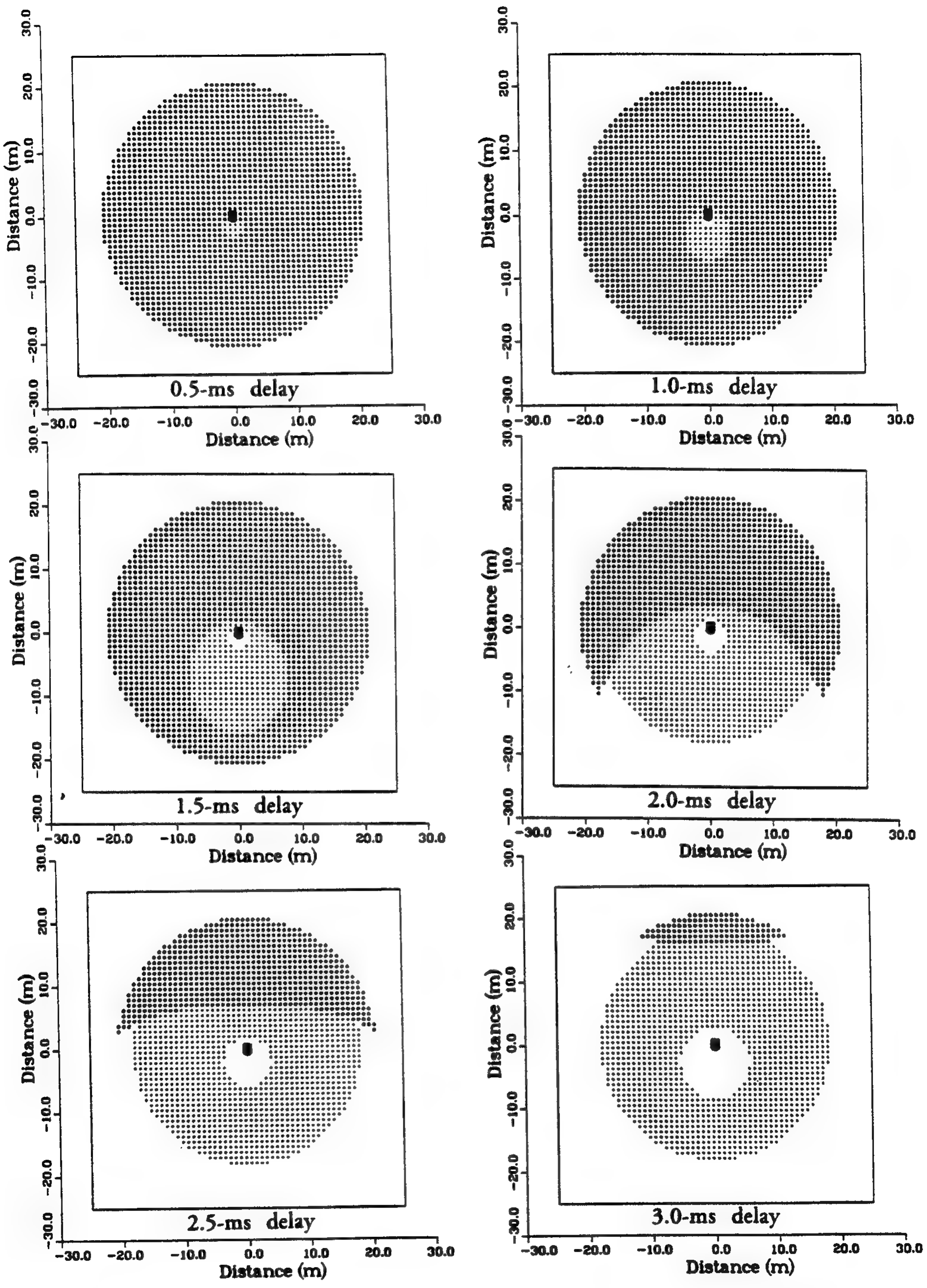


Figure 13. Coalescence maps for Zaker experiment simulations with three equal charges.

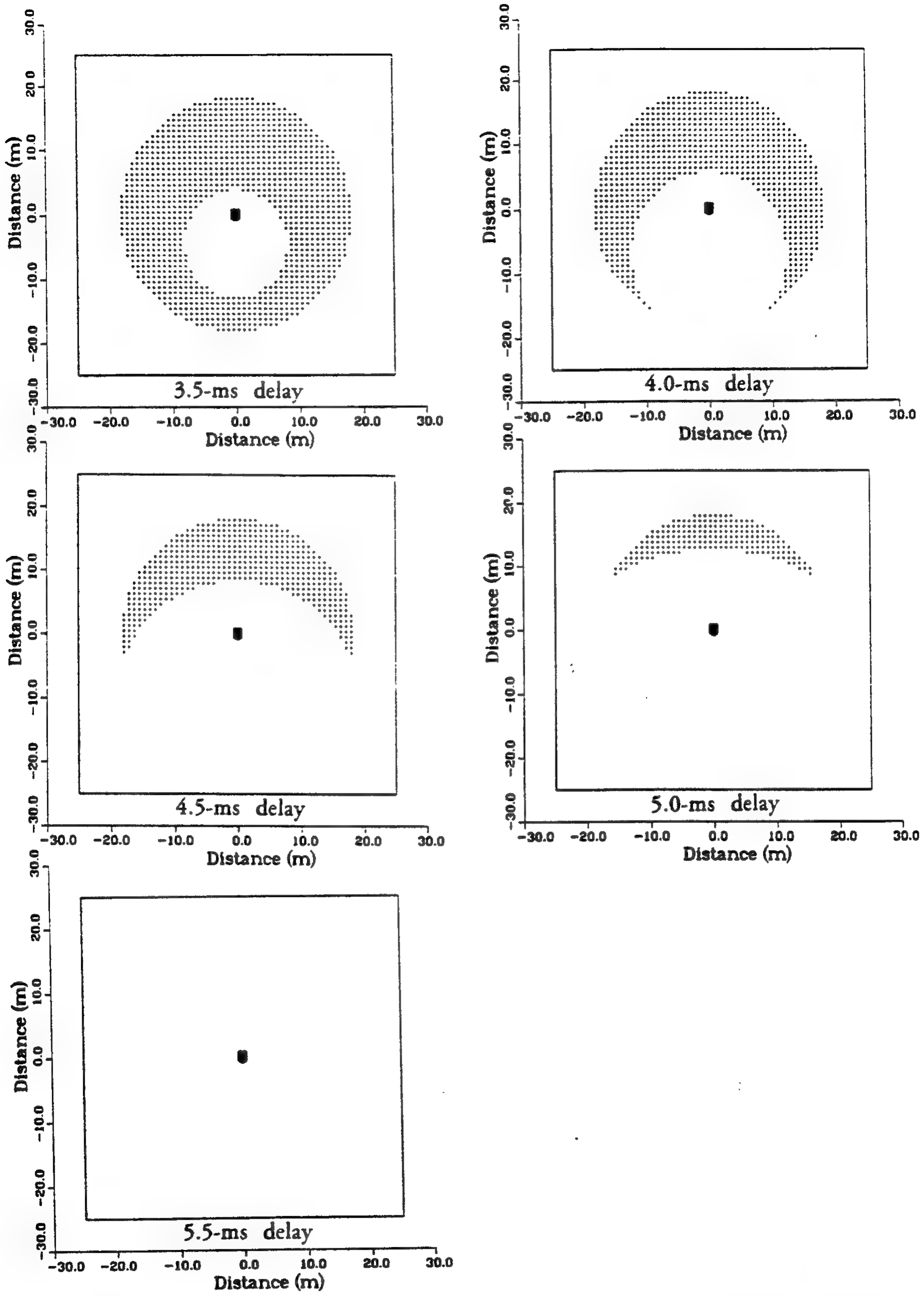


Figure 13. Coalescence maps for Zaker experiment simulations with three equal charges (continued).

The coalescence patterns produced in simulations of two equal charges are shown in Figure 14. As expected, symmetry is produced, with the single-donor coalescence regions (shown in Figure 9) reflected across the horizontal axis.

The coalescence regions produced with two unequal charges are not symmetrical about the horizontal axis as shown in Figure 15. These are combinations of the single-donor regions (shown in Figure 10) with reflections of the single-donor regions (shown in Figure 11) about the horizontal axis.

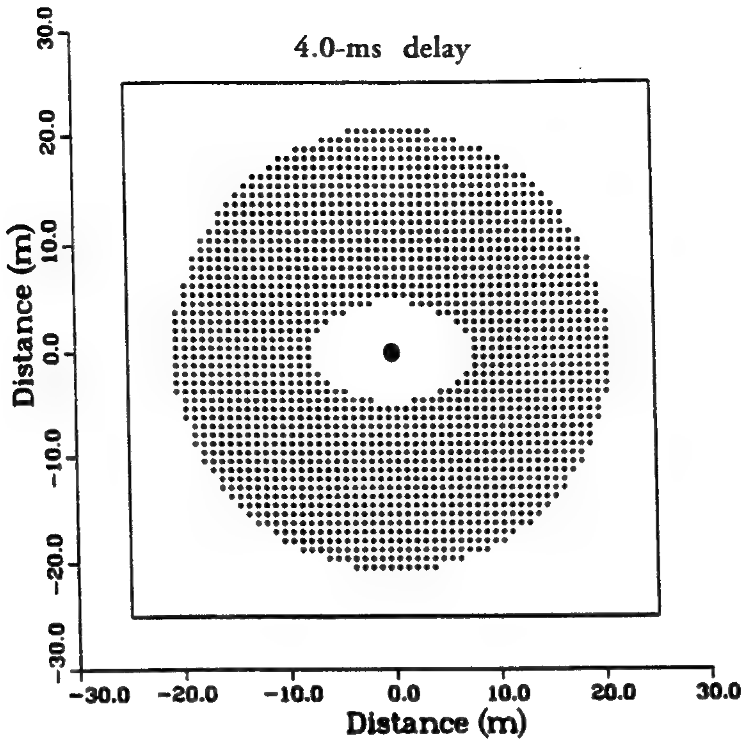
Symmetry about the horizontal axis also results with three equal charges, as shown in Figure 16. In this case, however, the region of coalescence is greater than that associated with reflection of the single donor regions (shown in Figure 13) because coalescence produced by the center-charge donor dominates. In fact, coalescence of the two waves from the outer charges persists at all delays since they detonate simultaneously when the central charge is the donor.

6. PERSONAL COMPUTER VERSION GRAPHICS

When BWACO was adapted for use on the personal computer, the coalescence map was enhanced with the addition of contours of the total explosive weight associated with the coalesced waves and a capability for producing peak overpressure contour plots was added. Examples of these maps are shown in Figure 17. The maps may be obtained either on the PC display or in hard-copy form.

Regions in which coalescence is detected are hatched in the coalescence map. Each contour is labeled to indicate the total explosive weight contributing to the worst-case blast environment within that contour. However, the outermost contour line is further limited by the overpressure limit for inhabited buildings. An inhabited building distance circle centered at the center of the charge is also shown. The legend lists all the contour levels present in the plot. The highest contour level represents the maximum effective explosive weight for the configuration.

In the peak overpressure map, contours are labeled to indicate the worst-case peak overpressure at each point within the region of interest. The contour levels are those specified in AR 385-64. The lowest level varies between 0.9 and 1.2 psi, depending on the total explosive weight. The limits of the blast hazard area determined in accordance with the regulations are also shown. The legend lists the stack positions



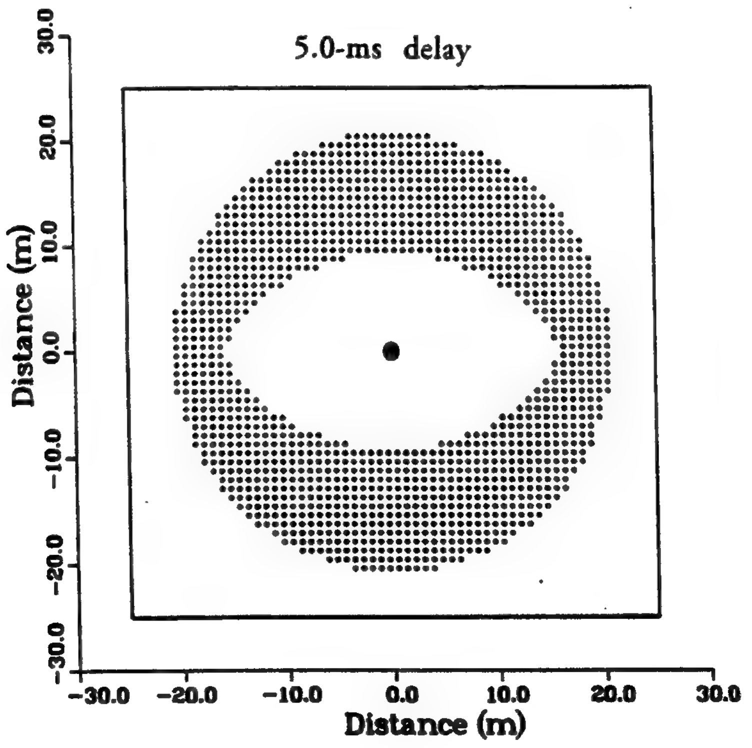


Figure 14. Coalescence maps for Zaker configuration simulations with two equal charges and multiple donors.

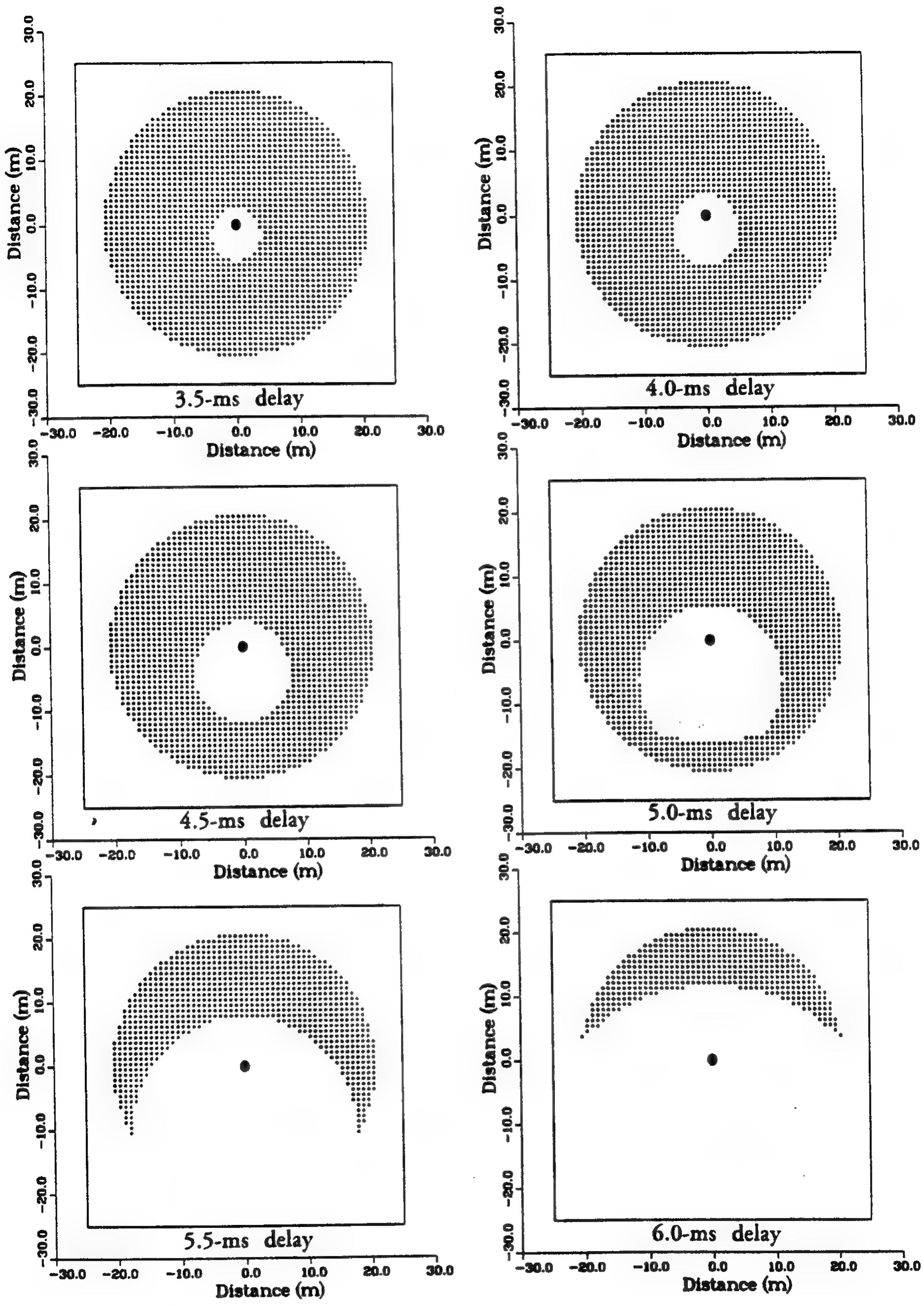


Figure 15. Coalescence maps for Zaker configuration simulations with two unequal charges and multiple donors.

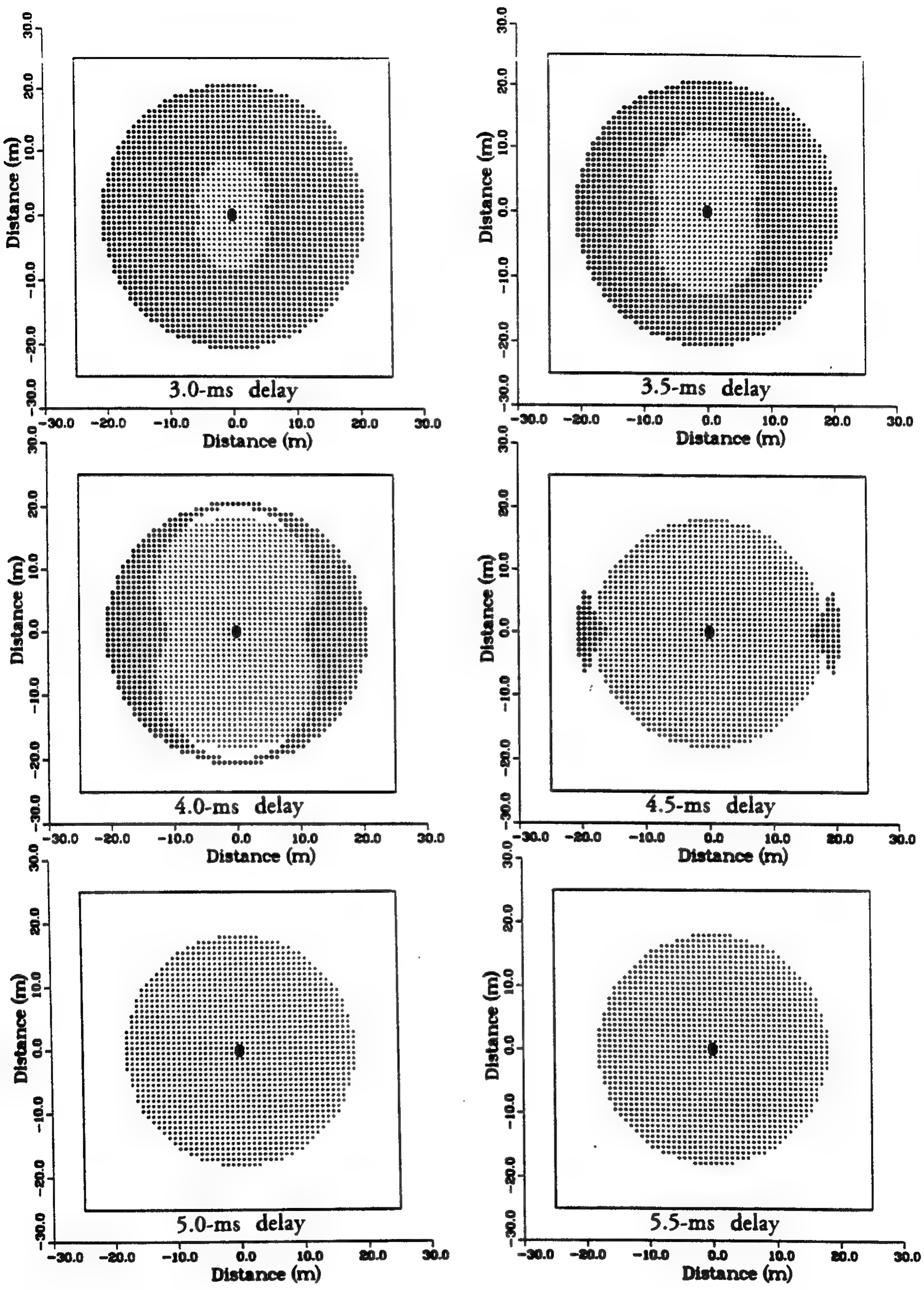
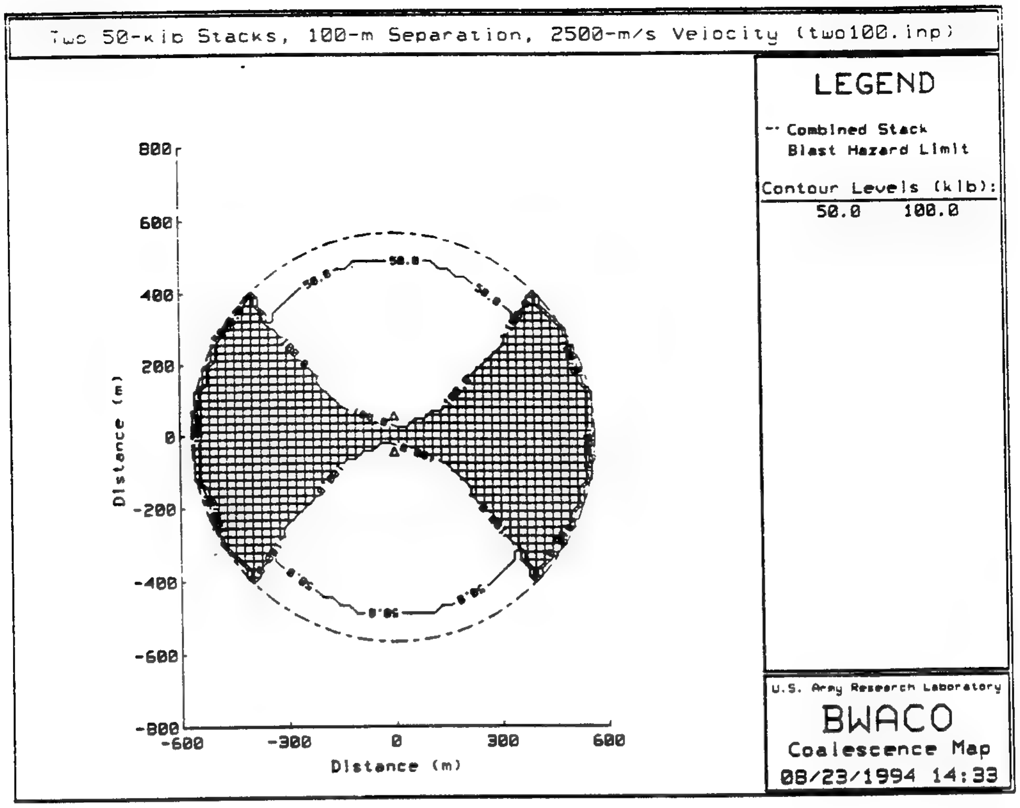


Figure 16. Coalescence maps for Zaker configuration simulations with three equal charges and multiple donors.



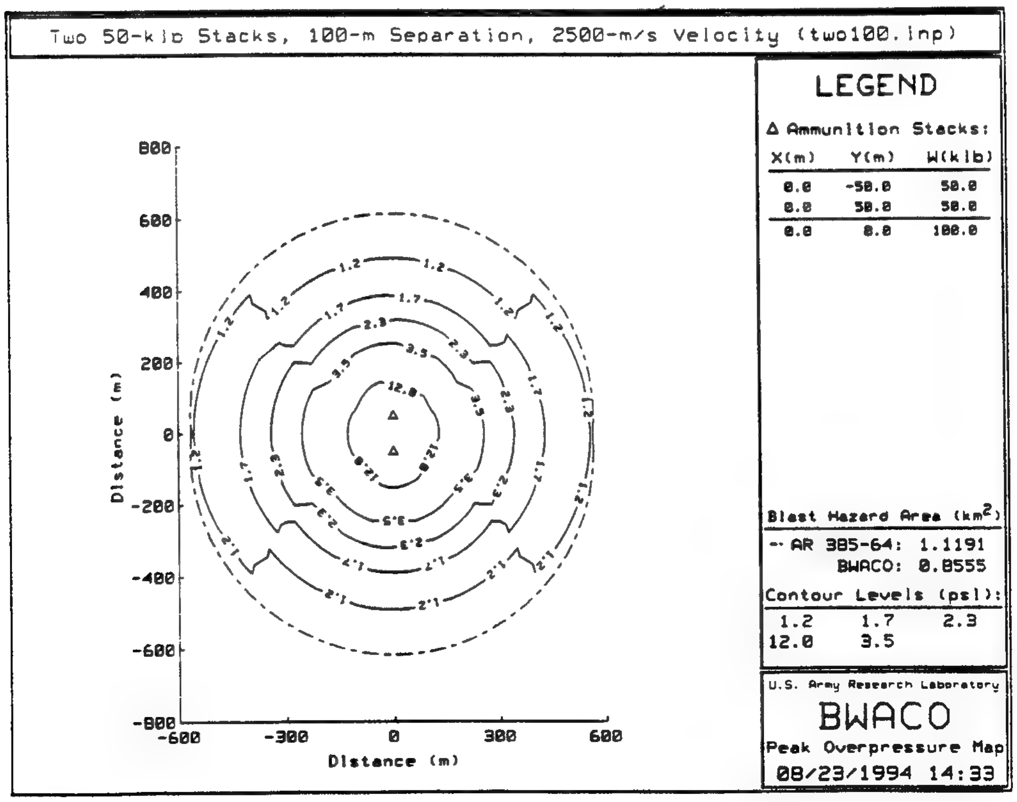


Figure 17. Coalescence and peak overpressure maps from the PC version of BWACO.

and weights, along with center of charge and total weight, compares the regulatory blast hazard with that determined by BWACO, and lists the contour levels.

7. SIMULATIONS OF REPRESENTATIVE LARGE-SCALE CONFIGURATIONS

In the large-scale scenarios that BWACO is intended to treat, initiation delay depends on the separation between stacks. The simulations described in the following paragraphs were intended to illustrate the effects of the separation between stacks and the distribution of explosive weight among stacks when all stacks act as initial donors. Most of them were made using a total equivalent weight of 100,000 lb distributed over two or more stacks. For this total weight, regulations permit an inhabited building exposure of 1.2 psi or less. The velocity of propagation between stacks was fixed at 2,500 m/s.

Results obtained with two stacks of 50,000 lb each are shown in Figures 18 and 19. At the smallest separation distance (50 m), the regulatory blast hazard region nearly coincides with the BWACO prediction. As the separation is increased, significant regions in which the pressure is below 1.2 psi appear. In these cases, requiring consideration of the blast produced by the total explosive weight of both stacks is clearly too restrictive. At the larger separation distances, regions of coalescence persist only near the horizontal symmetry axis. Here, consideration of the total weight is necessary. The variation of the blast hazard areas with separation is summarized in Table 1. The regulatory hazard area increases with separation while that predicted by BWACO is minimized somewhere between 100 and 150 m.

Table 1. Variation of Blast Hazard Areas with Separation for Two 50,000-lb Stacks

Separation (m)	Blast Hazard Area	
	AR 385-64 (km^2)	BWACO (km^2)
50	1.06	0.97
100	1.12	0.86
150	1.18	0.86
200	1.23	0.88

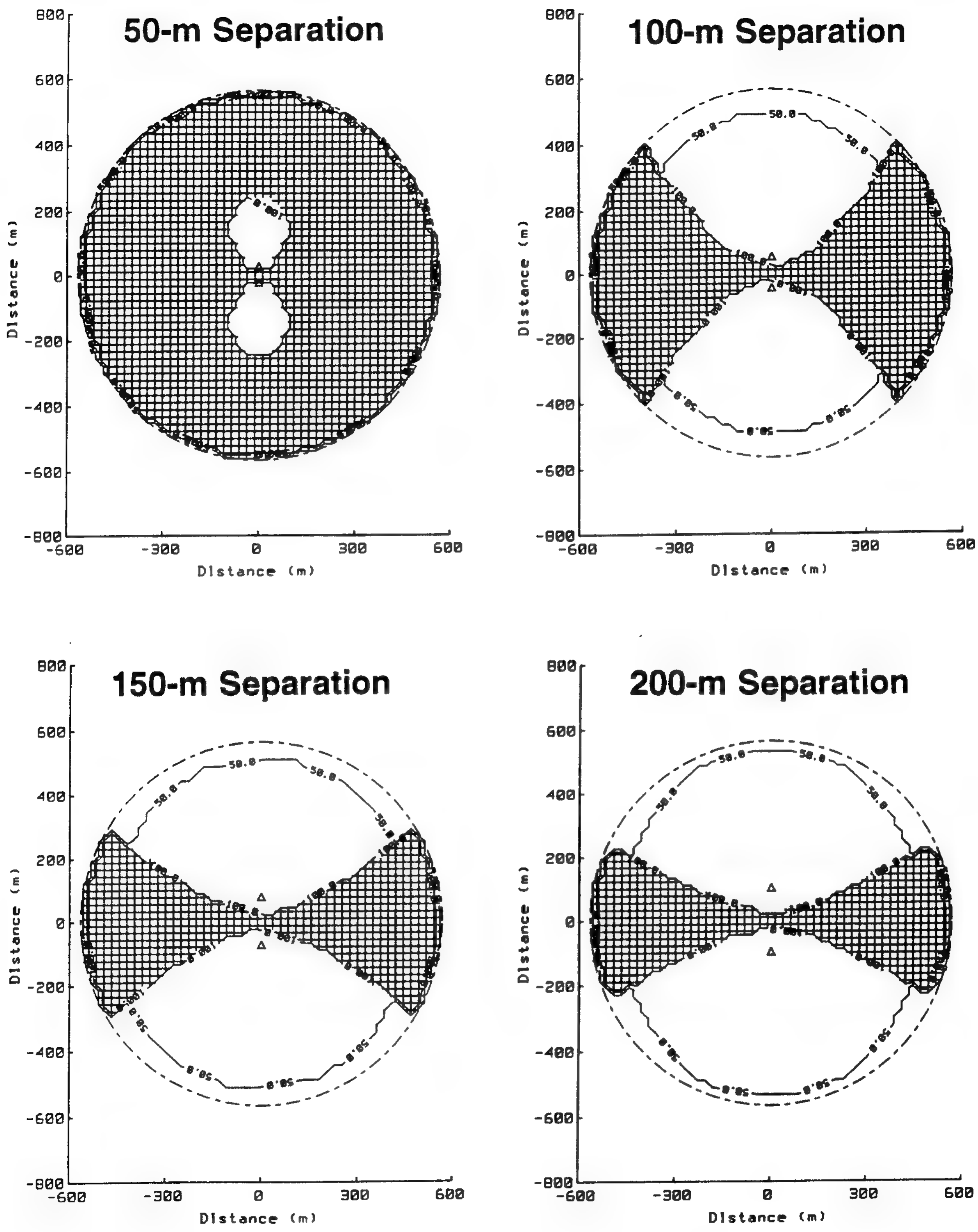


Figure 18. Coalescence maps for large-scale simulations with two 50,000-lb stacks.

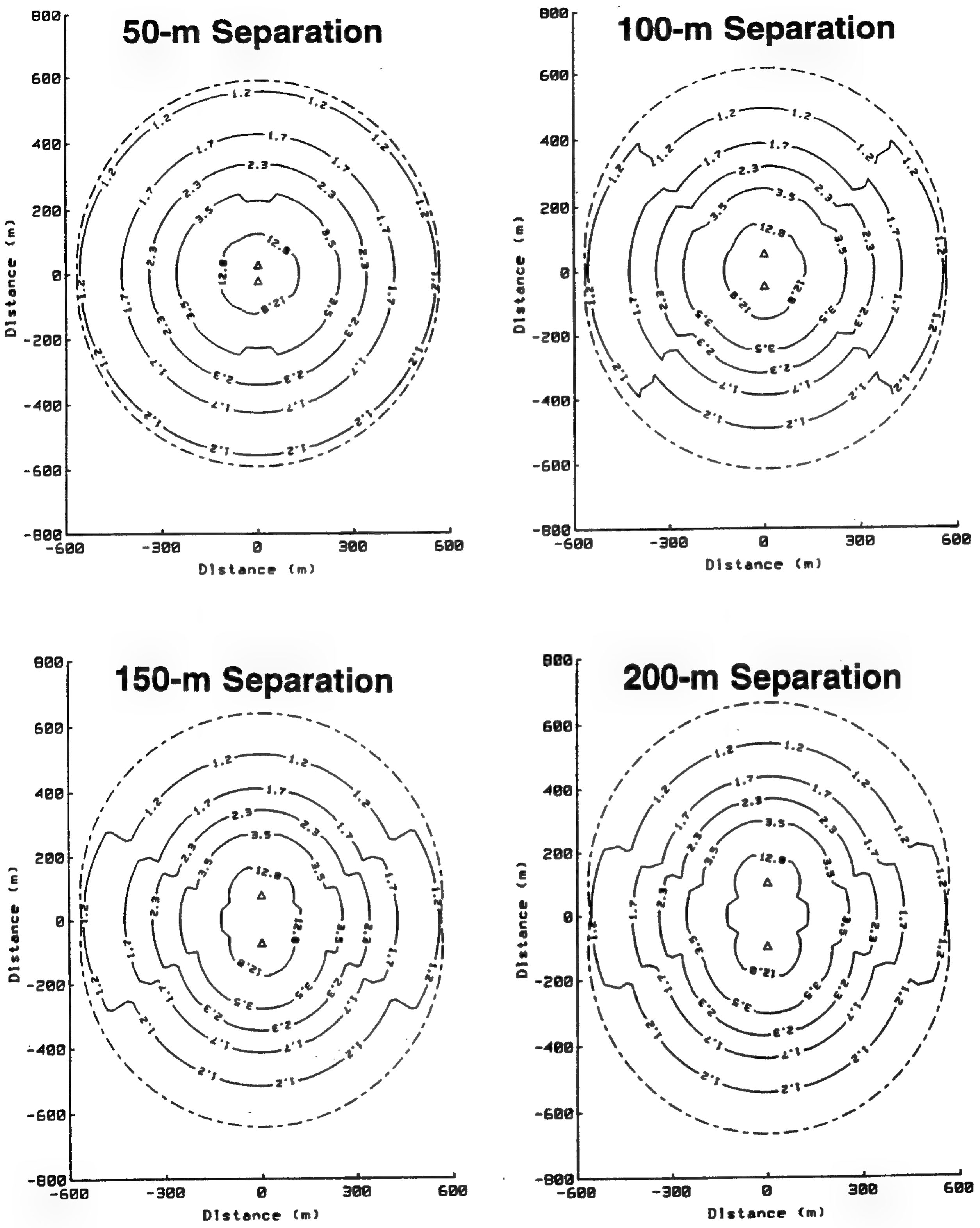


Figure 19. Peak overpressure maps for large-scale simulations with two 50,000-lb stacks.

We also simulated the effects of varying the distribution of explosive weight between two stacks separated by 150 m. The results, shown in Figures 20 and 21, indicate very little change in the coalescence region along the horizontal axis as the weight is redistributed. The hazard area increases slightly for increasingly unequal weight distributions as shown in Table 2.

Table 2. Variation of Blast Hazard Areas with Explosive Weight Distribution for Two Stacks Having a Total Weight of 100,000 lb Separated by 150 m

Weight Ratio (lb:lb)	Blast Hazard Area	
	AR 385-64 (km^2)	BWACO (km^2)
50,000 : 50,000	1.18	0.86
40,000 : 60,000	1.18	0.85
30,000 : 70,000	1.18	0.85
20,000 : 80,000	1.18	0.89
10,000 : 90,000	1.18	0.93

The results observed with four 25,000-lb stacks are similar to those seen with two 50,000-lb stacks. They are shown in Figures 22 and 23. Coalescence of waves from all four stacks and from three of the four stacks is only exhibited at the 50-m separation distance. These regions appear as small islands lying beyond the general regions of coalescence. At the 100-m separation, only two-stack coalescence results. As the separation is further increased, this coalescence becomes limited to the vicinity of the vertical and horizontal symmetry axes. The effect of separation on the blast hazard areas is given in Table 3. Again, BWACO predicts a minimum hazard near the 100-m separation.

Table 3. Variation of Blast Hazard Areas with Separation for Four 25,000-lb Stacks

Separation (m)	Blast Hazard Area	
	AR 385-64 (km^2)	BWACO (km^2)
50	1.13	0.85
100	1.25	0.71
150	1.37	0.73

The effect of the number of stacks was also investigated. If each stack in the 4-stack arrangement is divided into 4 equal stacks (6,250 lb ea) which are positioned such that their center of charge lies at the position of the original stack while retaining a uniform separation, an arrangement of 16 stacks with half the original separation results. The coalescence and peak overpressure maps produced are shown in Figures 24 and 25. Each map may be compared to the corresponding map in Figures 22 or 23. A significant advantage to smaller subdivisions is indicated. The effect of separation on the maximum effective explosive weight and the blast hazard areas is given in Table 4. Comparison with Table 3 shows a reduction in the predicted blast hazard area but an increase in the regulatory area. The minimum predicted hazard area occurs near the 50-m separation.

Table 4. Variation of Maximum Effective Explosive Weight and Blast Hazard Areas with Separation for Sixteen 6,250-lb Stacks

Separation (m)	Maximum Effective Explosive Weight (lb)	Blast Hazard Area: AR 385-64 (km^2)	Blast Hazard Area: BWACO (km^2)
25	56,250	1.18	0.64
50	25,000	1.37	0.50
75	25,000	1.57	0.56

Simulations of representative storage configurations with 6 and 10 unequal stacks demonstrate the utility of BWACO. In each case, the stacks have TNT equivalent weights between 20,000 and 30,000 lb and are separated from their neighbors by a distance of 100 m.

The six stacks have a total weight of 150,000 lb. The coalescence and peak overpressure maps are shown in Figure 26. The weight contours in the coalescence map are hard to distinguish but their levels are listed in the legend. The highest weight effective within the region of significant pressure is 99,000 lb. This is significantly less than the total weight. The level of the outermost contour shown on the peak overpressure map is 1.1 psi. This corresponds to the regulatory requirement for the 150,000-lb total weight. The blast hazard area predicted by BWACO using this pressure level is less than half that specified by the regulation. If the larger peak overpressure level corresponding to the maximum effective weight (99,000 lb) were used, an even more favorable comparison would result.

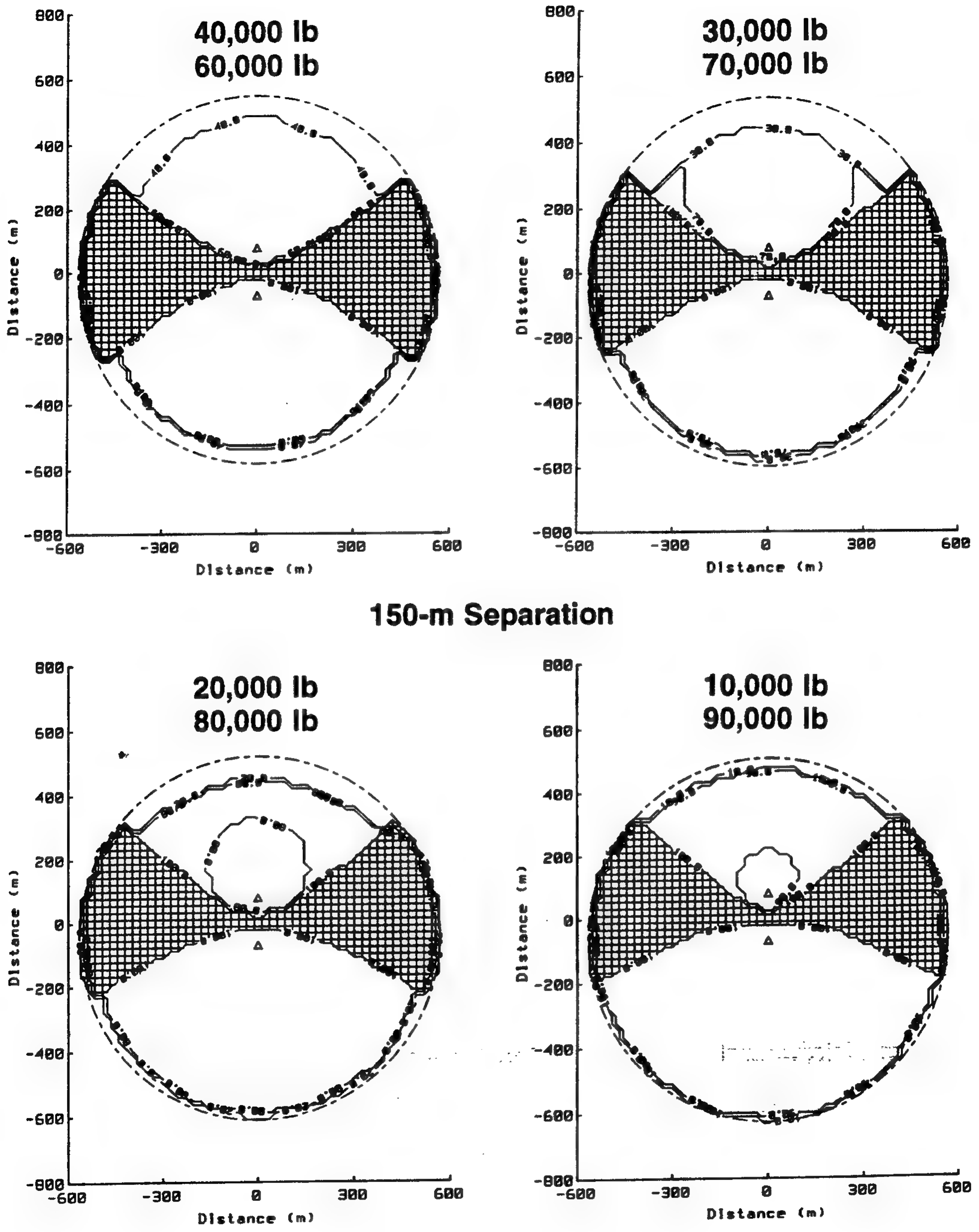


Figure 20. Coalescence maps for large-scale simulations with two unequal stacks having a total weight of 100,000 lb.

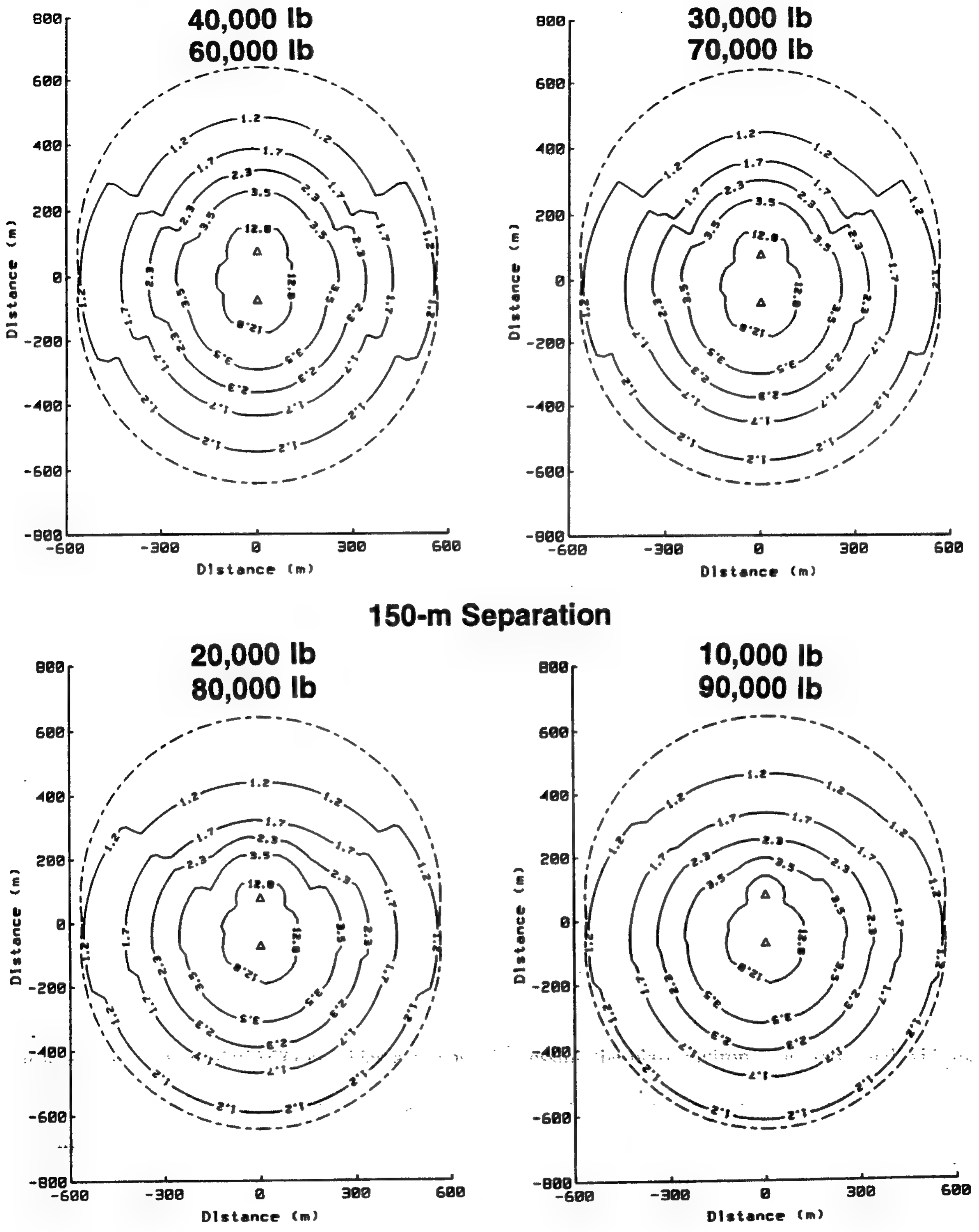


Figure 21. Peak overpressure maps for large-scale simulations with two unequal stacks having a total weight of 100,000 lb.

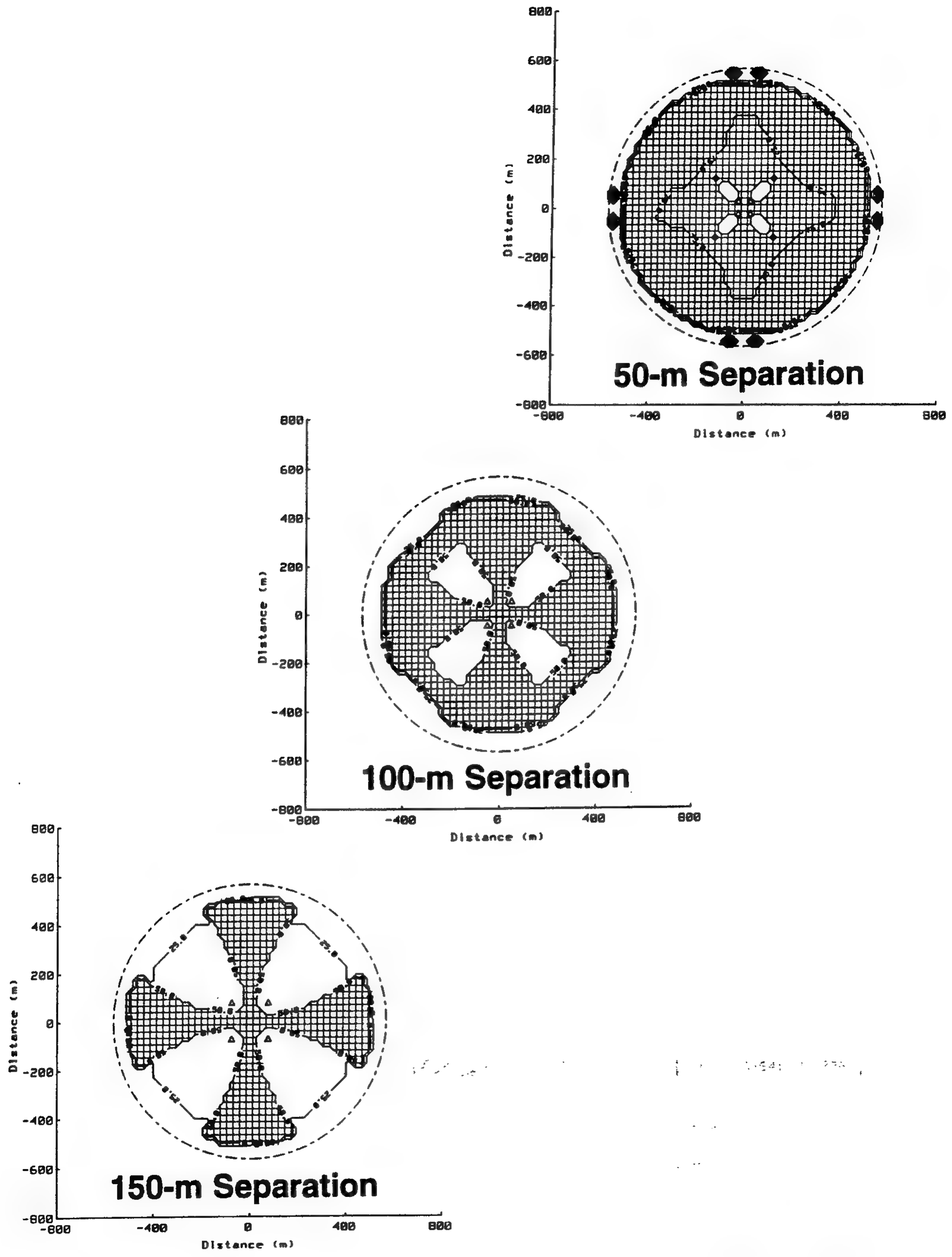


Figure 22. Coalescence maps for large-scale simulations with four 25,000-lb stacks.

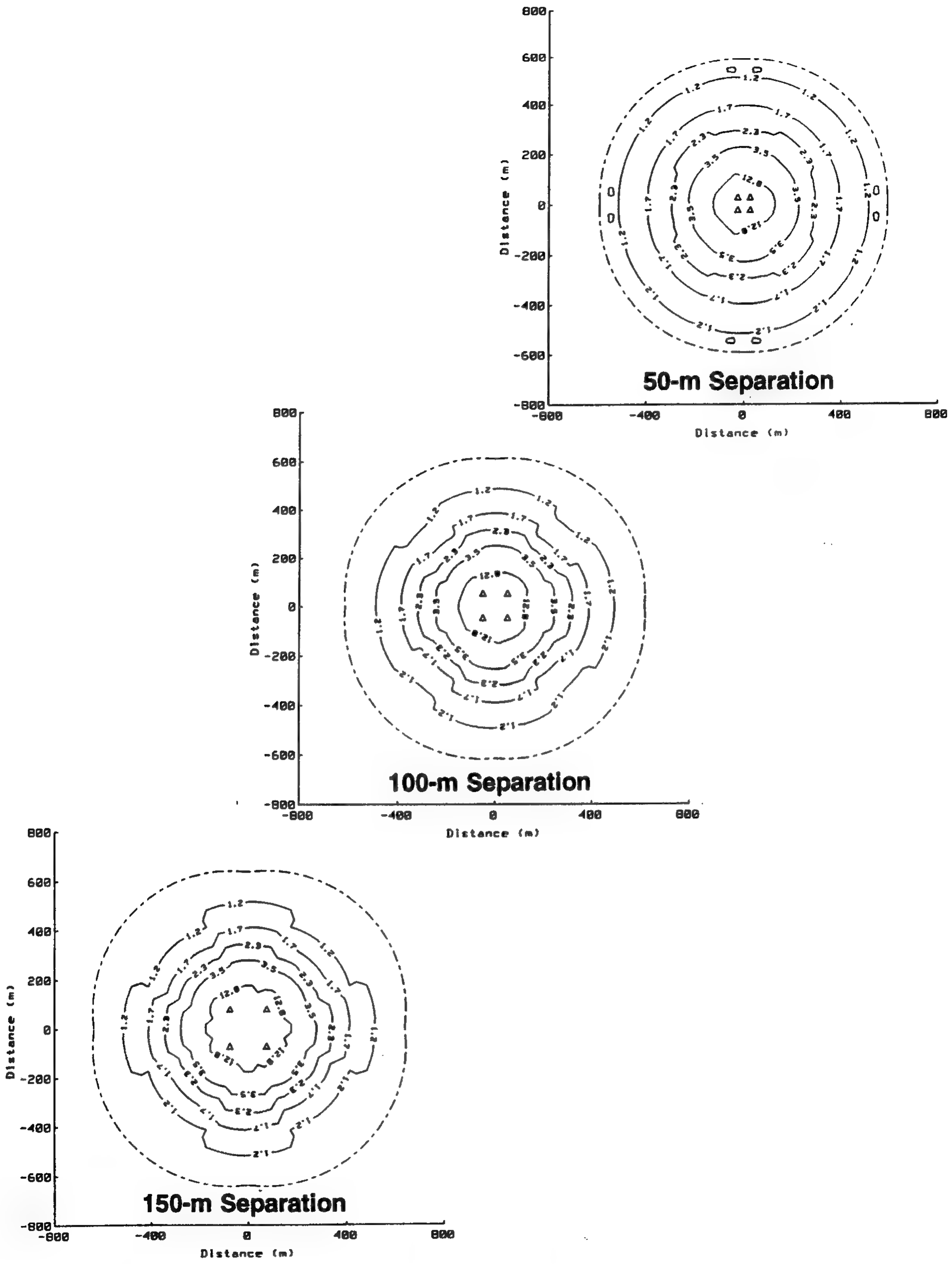


Figure 23. Peak overpressure maps for large-scale simulations with four 25,000-lb stacks.

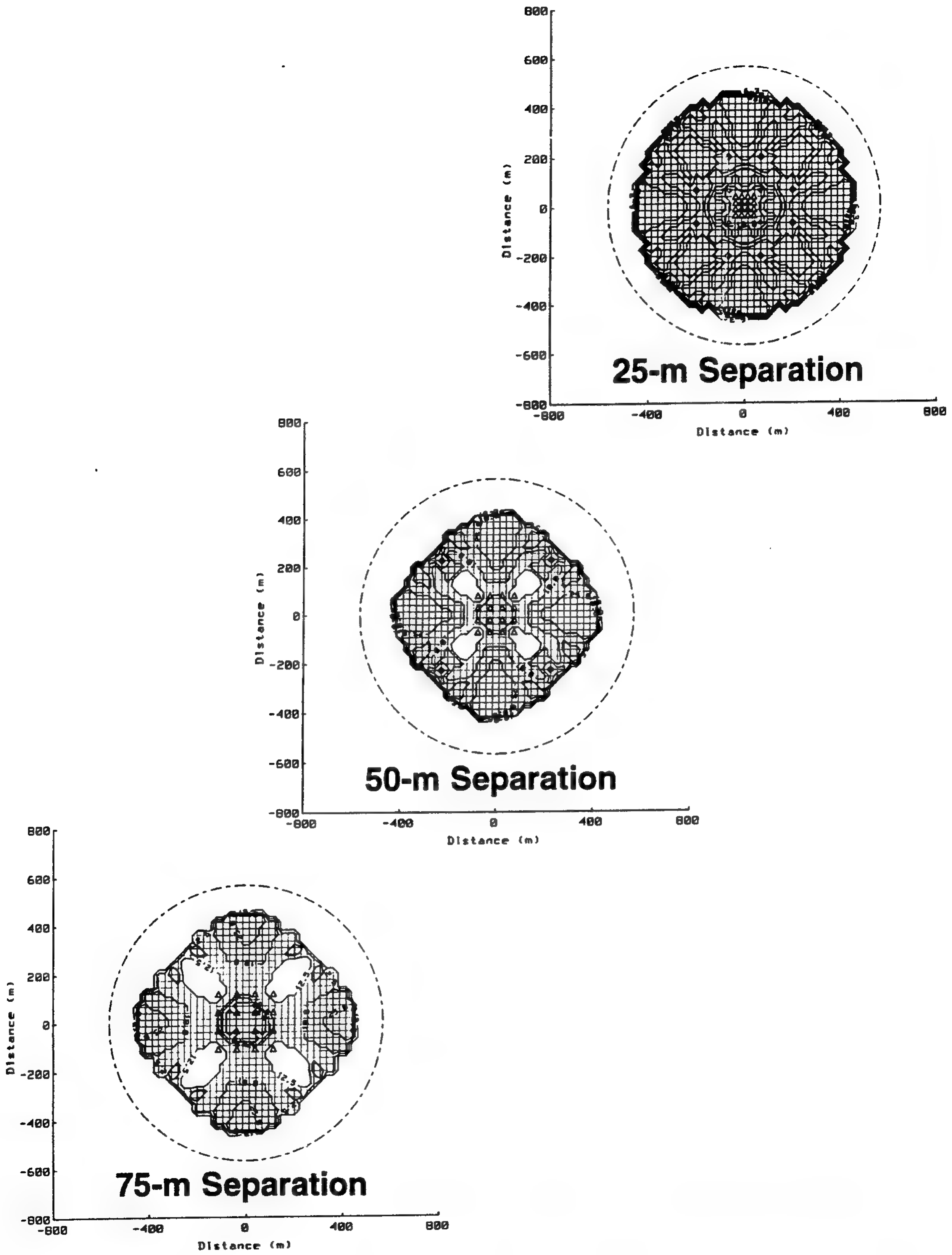


Figure 24. Coalescence maps for large-scale simulations with sixteen 6,250-lb stacks.

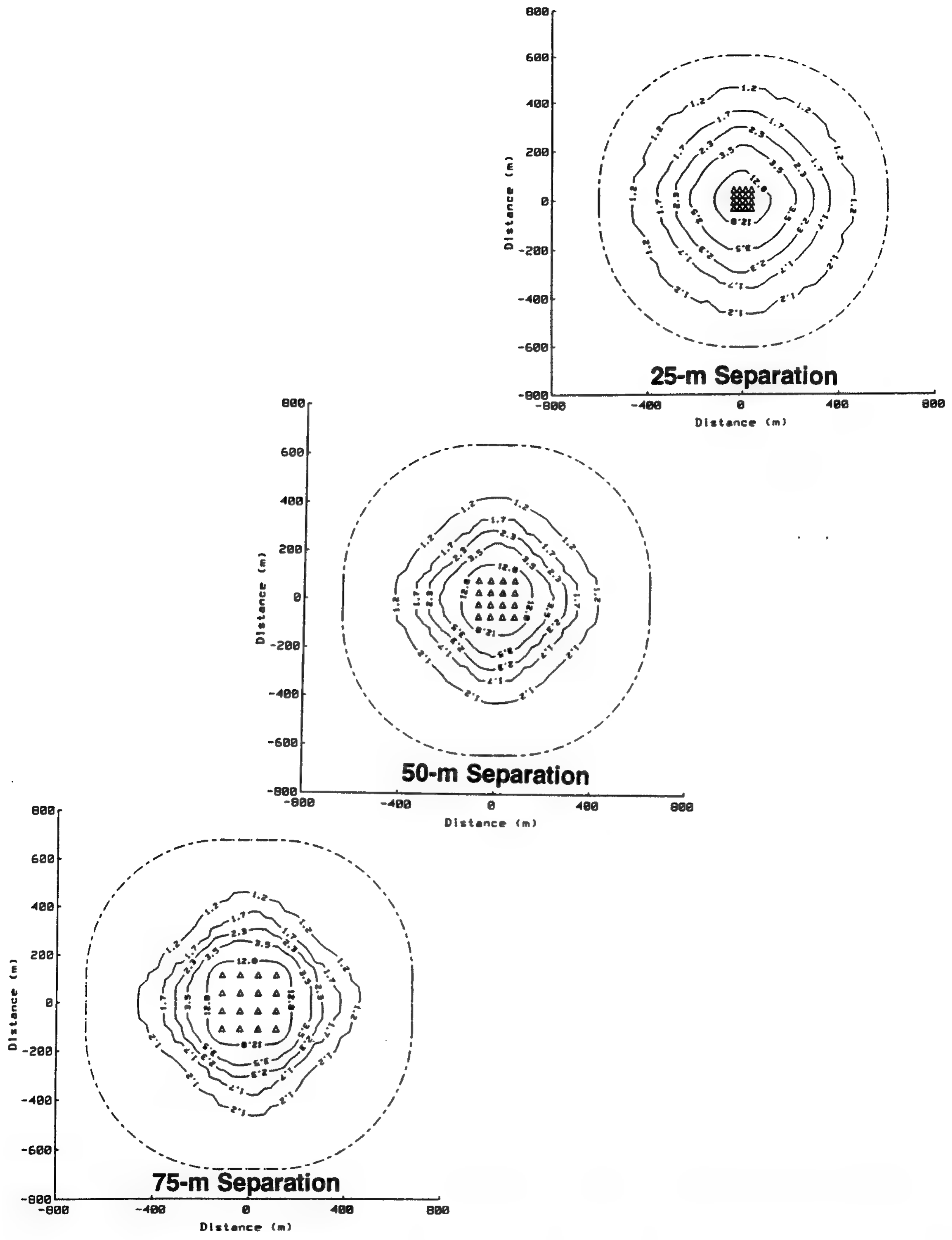
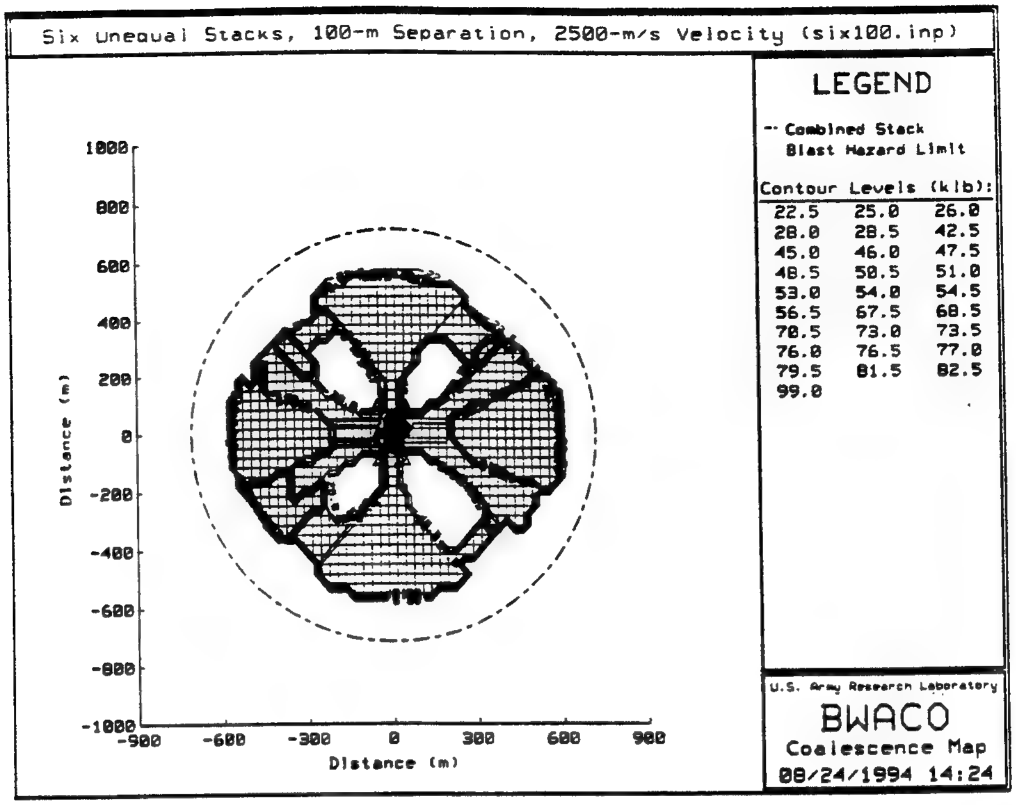


Figure 25. Peak overpressure maps for large-scale simulations with sixteen 6,250-lb stacks.



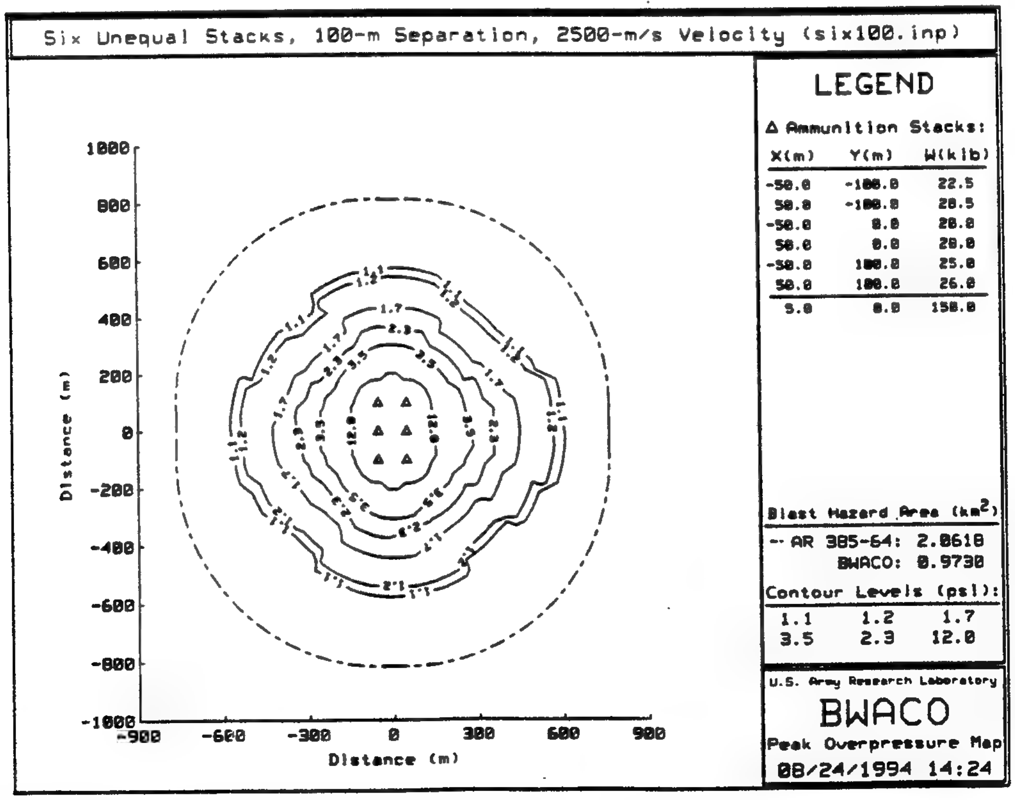


Figure 26. Coalescence and peak overpressure maps for a large-scale simulation with six unequal stacks having a total weight of 150,000 lb.

The ten stacks have a total weight of 250,000 lb, requiring consideration of peak overpressures down to 0.9 psi. The coalescence and peak overpressure maps are shown in Figure 27.

The highest weight effective within the region of significant pressure is 134,000 lb. The ratio of the BWACO hazard area to the regulatory hazard area is even smaller than in the six-stack case. Again, this comparison would become still more favorable if the larger overpressure corresponding to the maximum effective weight were used.

8. SUMMARY AND CONCLUSIONS

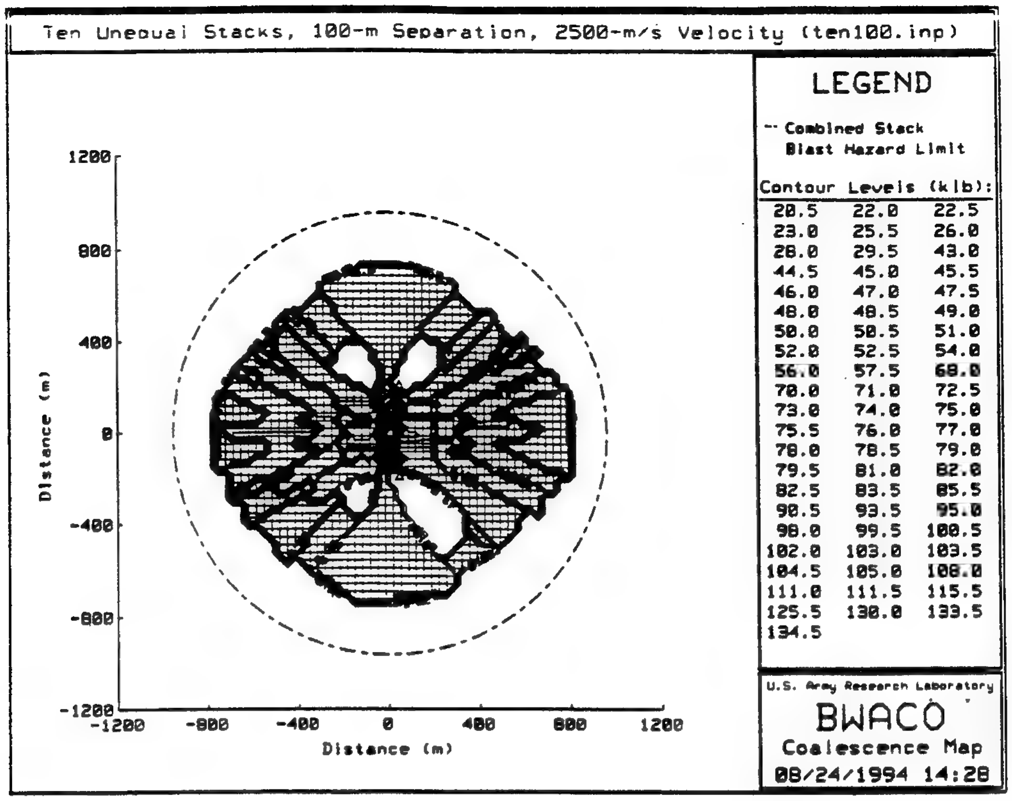
In this report, we have explained the assumptions underlying the BWACO algorithms, documented the evolution of BWACO based on comparisons with available experimental data, and demonstrated the application of BWACO to typical large-scale ammunition storage configurations.

Assumptions were made in order to determine the order and timing of the detonation of the stacks following the detonation of any initial donor stack, to establish a criterion for detecting coalescence, and to determine the combined pressure associated with a number of coalesced waves at a point.

Comparison of preliminary results with experimental data obtained by Zaker led to replacement of the standards initially used for the description of blast waves. Use of the 1-KT Standard was found to produce predictions which tended to miss detection of coalescence where the experiments showed that coalescence occurred. It was replaced with a standard based on experimental data reported by Kingery. In addition, a pressure combination algorithm based on Zaker's observation was found to produce more satisfactory results than other algorithms.

BWACO has been adapted for the personal computer with enhanced graphical representations. As currently configured, BWACO provides a means of assessing the blast environment associated with the sequential detonation of an arbitrary arrangement of ammunition stacks. The limitations imposed by the assumptions have not been assessed in realistic configurations.

Applications to a number of problems representative of typical ammunition storage configurations were detailed. The results indicate that regions of significant pressure associated with the coalescence of blast waves from distributed ammunition stacks may be less extensive than corresponding regions associated



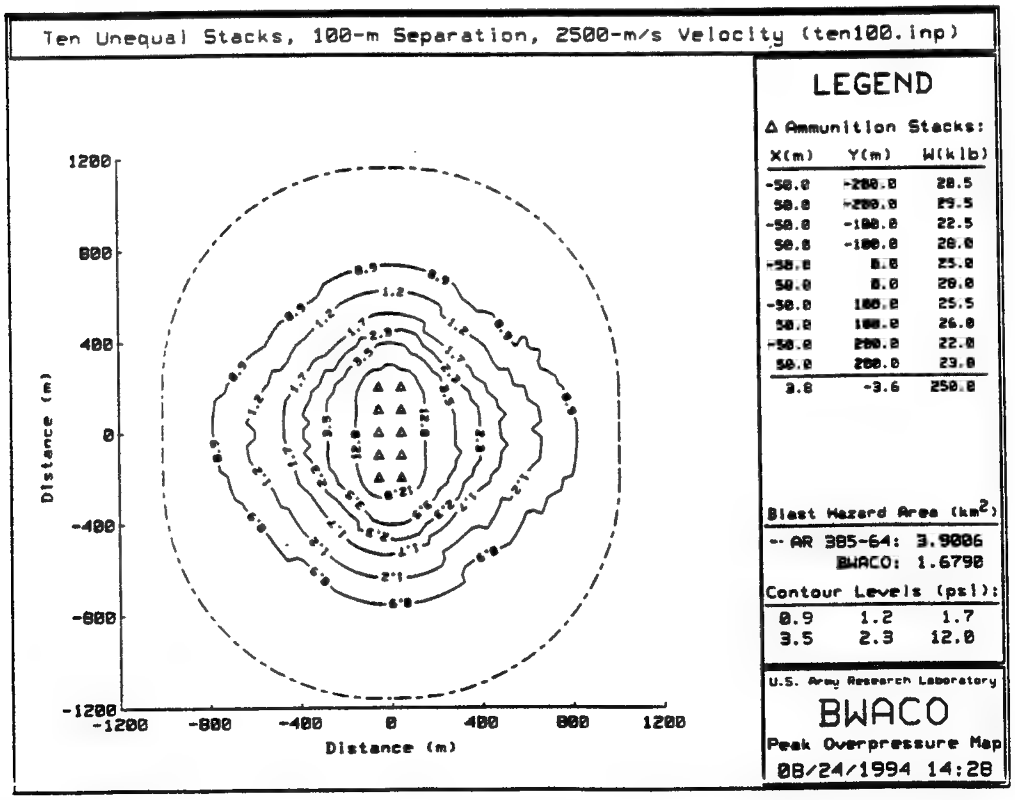


Figure 27. Coalescence and peak overpressure maps for a large-scale simulation with ten unequal stacks having a total weight of 250,000 lb.

with regulatory requirements. An advantage associated with the distribution of ammunition into smaller subdivisions was also demonstrated.

If it proves advantageous, opportunities to apply more sophisticated models can be exploited. Areas where improvement is possible include the propagation model and the pressure combination algorithm, as well as the addition of terrain effects and fragment hazard predictions.

INTENTIONALLY LEFT BLANK.

9. REFERENCES

Brode, H. L. "Quick Estimates of Peak Overpressure From Two Simultaneous Blast Waves." R&D Associates, RDA-TR-107006-008, Marina Del Rey, CA, December 1977.

Department of the Army. AR 385-64, "Ammunition and Explosives Safety Standards." 22 May 1987.

Hikida, S., and C. E. Needham. "Low Altitude Multiple Burst (LAMB) Model." S-Cubed Final Report S-CUBED-R-81-5067, 30 June 1981.

Kingery, C. N., J. H. Keefer, and J. D. Day. "Surface Air Blast Measurements From a 100-Ton TNT Detonation." BRL Memorandum Report No. 1410, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, June 1962.

Kingery, C. N., and B. F. Pannill. "Peak Overpressure vs. Scaled Distance for TNT Surface Bursts (Hemispherical Charges)." BRL Memorandum Report No. 1518, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, April 1964.

Kingery, C. N. "Air Blast Parameters vs. Distance for Hemispherical TNT Surface Bursts." BRL Report No. 1344, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, September 1966.

Needham, C. E., and J. E. Crepeau. "The DNA Nuclear Blast Standard (1 KT)." Systems, Science and Software Report, SSS-R-81-4845, 30 January 1981.

Zaker, T. A. "Far Field Overpressure From Closely Spaced Sequential Detonations." Minutes of the Eleventh Explosive Safety Seminar, 9-10 September 1969.

INTENTIONALLY LEFT BLANK.

APPENDIX A:

DETAILED DESCRIPTION OF BWACO SOURCE CODE

INTENTIONALLY LEFT BLANK.

BACKGROUND

The BWACO program was written to predict regions of blast wave coalescence associated with the sequential mass detonation of distributed ammunition stacks positioned as specified by the user. It accomplishes this by assuming that blast waves propagate independently of one another and that coalescence occurs wherever and whenever the shock associated with one blast wave encroaches into the positive overpressure phase of another blast wave. It further assumes that communication of mass detonation between stacks is controlled by a user-supplied propagation velocity (which might represent the velocity of fragments from the donor stack). It uses this propagation velocity to determine the order and timing of detonation of the stacks. It recognizes that the identity of the initial donor is not generally known and, unless a single donor has been specified by the user, considers all possible donors in order to predict the worst-case environment at any point. It reports the peak overpressure and effective explosive weight contributing to this environment throughout the region of interest.

Although the vast majority of BWACO represents new code, a few of its FORTRAN subroutines and functions were adapted from the DNA Nuclear Blast Standard (1 KT) (hereinafter referred to as the 1-KT Standard), which describes the blast environment produced by a 1-KT nuclear explosion, and from the Low Altitude Multiple Burst (LAMB) model, which describes the blast environment produced by multiple nuclear explosions of arbitrary yield. As a result, much of the terminology used to discuss the program comes from the nuclear blast vocabulary. For example, the net equivalent explosive weight in an ammunition stack is referred to as its yield and detonation of a stack is called a burst.

The organization of BWACO's subroutines and functions is diagrammed in Figure A-1 (where remaining adaptations from the 1-KT Standard or LAMB are rendered white on black). The routines are divided into six modules. The interface module directs BWACO's major activities, configuring the user session, setting the constants, and interfacing with the user by providing a menu of options. These include problem setup and execution, plotting, and reconfiguration of the session. The execution module reads the user's input, determines the region of interest, distributes stations within it, and executes the computation. The plotting module interfaces with the user to produce plots on request. The propagation module determines the order and timing of detonation propagation through the stacks. The coalescence module determines the waveform variables and applies the coalescence criterion at each station. The waveform module computes the required scale factors and determines the required waveform parameters from the applicable standard.

BWACO SUBROUTINES

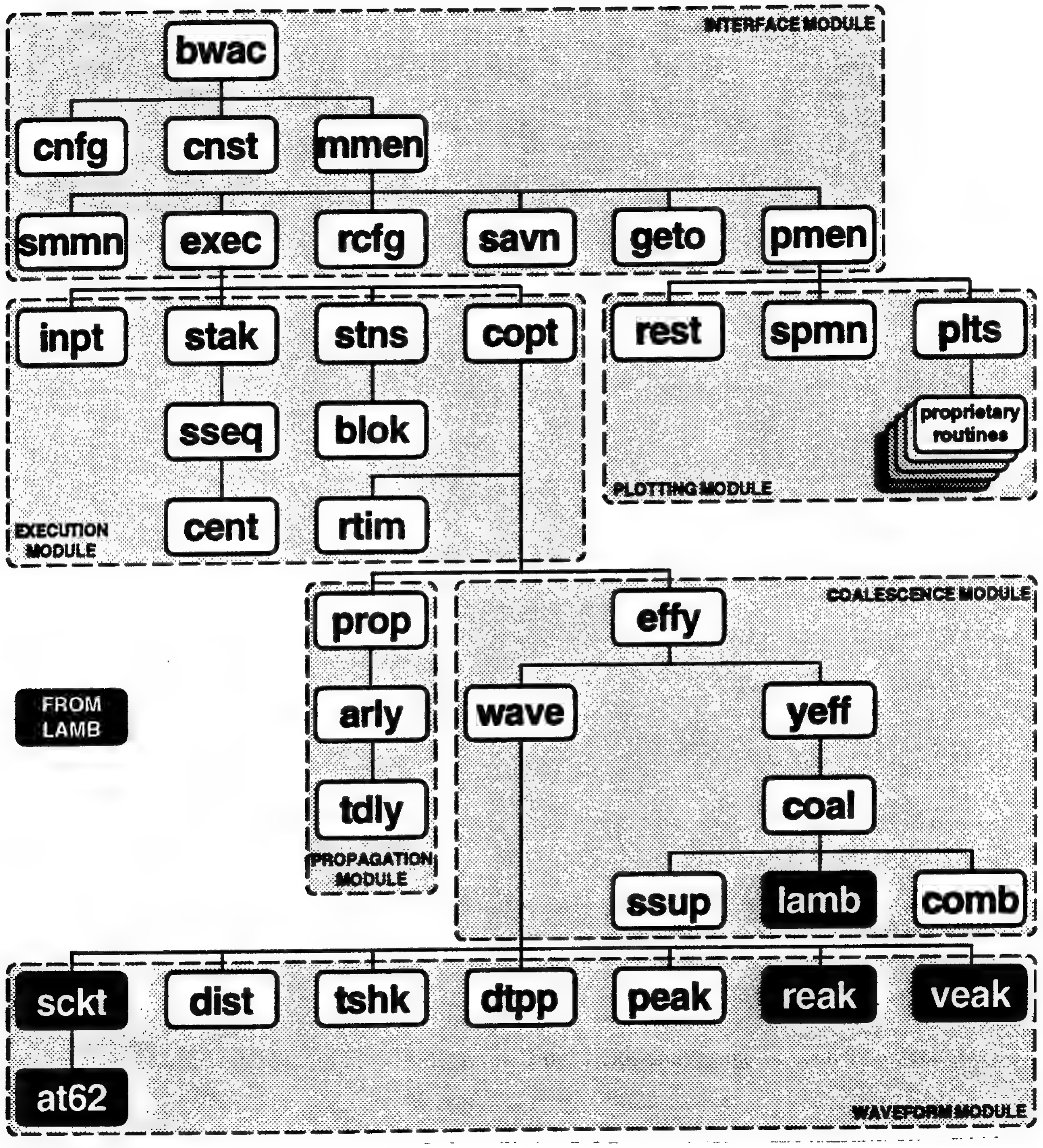


Figure A-1. Schematic of BWACO subroutine calling relationships.

The following discussion details most of the routines, describing the program from the inside out. BWACO's logic is difficult to understand and explain and the inside-out approach facilitates exposition of its operation.

WAVEFORM MODULE

These routines use the applicable blast standard to describe the shock wave, the boundary between the positive and negative phases of the blast wave (point of zero overpressure) and the peak values of overpressure, density, and particle velocity associated with a burst.

Subroutine *at62*, adapted from the 1-KT Standard, is shown in Figure A-2. It computes ambient pressure (Pa), sound speed (cm/s), density (g/cm^3) and temperature (K) from the 1962 U.S. standard temperature atmosphere for a given altitude (cm).

Subroutine *sckt*, adapted from LAMB, is shown in Figure A-3. It computes scale factors for time, distance and pressure for use with the 1-KT Standard given the altitude (cm) and yield (kt) of the burst and the altitude (cm) of the station of interest. Unneeded scale factors have been eliminated from this routine.

Function *dist* is shown in Figure A-4. It computes the radius from a stack to a station given their coordinates.

Function *tshk* is shown in Figure A-4. It computes the time of arrival of the shock wave produced by a 1-KT burst given the scale radius from the center of burst and scales it to the yield of the specified burst.

Function *dtpp* is shown in Figure A-4. It computes the positive phase duration of the blast wave produced by a 1-KT burst given the scale radius from the center of burst and scales it to the yield of the specified burst.

Function *peak* is shown in Figure A-5. It computes the peak overpressure of the blast wave associated with a specified burst given the scale radius from the center of burst.

```
c_______________________________________________________________________
c
      subroutine at62(tty,wsp,cs,wsr,wst)
c
c      computes standard atmosphere conditions at a given altitude
c      (adapted from DNA 1-KT STANDARD)
c
c        tabat(1) = r, the gas constant in erg/mole/K
c        tabat(2) = radius of the earth in cm
c        tabat(3) = acceleration due to gravity at sea level in cm/s/s
c        tabat(4) = molecular weight of air at sea level
c
c        tabz = altitude in cm
c        tabt = molecular scale temperature in K
c        tabl = molecular scale temperature gradient in K/cm
c        tabp = pressure in dyne/cm**2
c
c        nz = the number of altitudes
c
      dimension tabat(4),tabz(22),tabl(21),tabt(22),tabp(22)
c
c      data for temperate atmosphere
      data nz/     21/, rhoz/         1.22500000e-03/, tabat/
     + 8.3144000e+07,6.367488e+08,9.80665000e+02,2.8964400e+01/
      data tabz/
     + 0.            , 1.10190000e+06, 2.00630000e+06, 3.21620000e+06,
     + 4.73500000e+06, 5.24290000e+06, 6.15910000e+06, 7.99940000e+06,
     + 9.00000000e+06, 1.00000000e+07, 1.10000000e+07, 1.20000000e+07,
     + 1.50000000e+07, 1.60000000e+07, 1.70000000e+07, 1.90000000e+07,
     + 2.20000000e+07, 3.00000000e+07, 4.00000000e+07, 5.00000000e+07,
     + 6.00000000e+07, 7.00000000e+07/
      data tabl/
     +-6.49291769e-05, 9.28018576e-08, 9.86255062e-06, 2.77080373e-05,
     +-1.72248474e-07,-1.96000240e-05,-3.91696897e-05, 1.60821507e-07,
     + 2.98166740e-05, 5.02020080e-05, 9.97762300e-05, 2.00108809e-04,
     + 1.49589024e-04, 1.00407490e-04, 6.97598500e-05, 6.68801467e-05,
     + 3.49035000e-05, 3.31099360e-05, 2.58868500e-05, 1.71252960e-05,
     + 1.09162420e-05/
      data tabt/
     + 2.88150000e+02, 2.16604540e+02, 2.16688470e+02, 2.28621170e+02,
     + 2.70704137e+02, 2.70616652e+02, 2.52659110e+02, 1.80575130e+02,
     + 1.80736048e+02, 2.10552722e+02, 2.60754730e+02, 3.60530960e+02,
     + 9.60857386e+02, 1.11044641e+03, 1.21085390e+03, 1.35037360e+03,
     + 1.55101404e+03, 1.83024204e+03, 2.16134140e+03, 2.42020990e+03,
     + 2.59146286e+03, 2.70062528e+03/
      data tabp/
     + 1.01325000e+06, 2.26320000e+05, 5.47486994e+04, 8.68013979e+03,
     + 1.10904998e+03, 5.90004987e+02, 1.82098959e+02, 1.03769924e+01,
     + 1.64379881e+00, 3.00749781e-01, 7.35439451e-02, 2.52169805e-02,
     + 5.06169601e-03, 3.69429709e-03, 2.79259780e-03, 1.68519867e-03,
     + 8.67381898e-04, 1.94317430e-04, 4.15743157e-05, 1.13023464e-05,
     + 3.55894454e-06, 1.22936355e-06/
      data cons/1.389779604e14/
c
c      find the table index corresponding to the given altitude
      if(tty.gt.0.0) then
        do 110 jat=1,nz
          if(tty-tabz(jat)) 120,130,110
  110   continue
        jat=nz+1
  120   jat=jat-1
        jat=max0(1,jat)
      else
        jat=1
      endif
c
c      compute the standard atmosphere conditions
  130 dum2=(tabz(jat)-tty)/(tabat(2)+tty)/(tabat(2)+tabz(jat))
      dum3=(tabat(2)+tabz(jat))/(tabat(2)+tty)
      var1=tabat(2)*tabl(jat)-tabt(jat)+tabl(jat)*tabz(jat)
      var2=(tabt(jat)-tabl(jat)*tabz(jat)+tabl(jat)*tty)/tabt(jat)
      fs=-cons*(dum2/var1+tabl(jat)/(var1*var1)*alog(dum3*var2))
      wsp=tabp(jat)*exp(fs)
      wst=tabt(jat)+tabl(jat)*(tty-tabz(jat))
      wsr=wsp*tabt(1)*rhoz/(wst*tabp(1))
      cs=sqrt(1.4*wsp/wsr)
c
      return
c
      end
c_______________________________________________________________________
```

Figure A-2. Subroutine *at62* from the waveform module.

```
c_________________________________________________________________
c
      subroutine sckt(zbur,wbur,zsta, tscal,rscal,pscal)
c
c      computes scale factors for use with 1-kt standard
c       (adapted from DNA 1-KT STANDARD)

c       independent variables:
c        zbur = burst altitude in cm above sea level
c        zsta = station altitude in cm above sea level
c        wbur = yield of the burst in kt
c
c       dependent variables:
c        tscal = scales actual times to 1-kt sea level
c        rscal = scales 1 kt sea level dimensions to actual
c        pscal = scales 1 kt sea level pressures to actual
c
      common/conf/ thrd,tpi
      common/cona/ p1,c1,r1,t1
c
      call at62 (zsta, p3,c3,r3,t3)
      pscal=p3/p1
      if((zsta-zbur).ge.100.) then
        call at62(zbur, p2,c2,r2,t2)
        dz=zsta-zbur
        b=alog(p3/p2)/dz
        bzsta=b*zsta
        a=p3*exp(-bzsta)
        p3=a*(exp(bzsta)-exp(b*zbur))/(b*dz)
        b=alog(r3/r2)/dz
        bzsta=b*zsta
        a=r3*exp(-bzsta)
        r3=a*(exp(bzsta)-exp(b*zbur))/(b*dz)
        c3=sqrt(1.4*p3/r3)
      endif
      rscal=(wbur*p1/p3)**thrd
      tscal=c3/(rscal*c1)
c
      return
c
      end
c_________________________________________________________________
```

Figure A-3. Subroutine *sckt* from the waveform module.

```
c
c-----------------------------------------------------------------------
      function dist(xstan,ystan,zstan,xstck,ystck,zstck)
c
c      computes the radius from a stack to a station
c
c      independent variables:
c       xstan = x-coordinate of the station
c       ystan = y-coordinate of the station
c       zstan = z-coordinate of the station
c       xstck = x-coordinate of the stack
c       ystck = y-coordinate of the stack
c       zstck = z-coordinate of the stack
c
c      local variables:
c       deltx = x-component of separation between station and stack
c       delty = y-component of separation between station and stack
c       deltz = z-component of separation between station and stack
c
      deltx=xstan-xstck
      delty=ystan-ystck
      deltz=zstan-zstck
      dist=sqrt(deltx*deltx+delty*delty+deltz*deltz)
c
      return
c
      end
c-----------------------------------------------------------------------
c
c-----------------------------------------------------------------------
      function tshk(tscal,rslkt)
c
c      computes the shock arrival time for a
c      specified burst as a function of radius
c
c      independent variables:
c       tscal = time scale factor
c       rslkt = 1 kiloton scale radius
c
c      dependent variable:
c       tshk = shock arrival time
c
c      local variable:
c       tslkt = 1 kiloton scale arrival time
c
      common/cons/ r0s,c0s,d0s
c
c      compute the 1-kt shock arrival time
      tslkt=(rslkt-d0s*tanh(rslkt/r0s))/c0s
c
c      scale the arrival time to actual
      tshk=tslkt/tscal
c
      return
c
      end
c-----------------------------------------------------------------------
c
c-----------------------------------------------------------------------
      function dtpp(tscal,rslkt)
c
c      computes the positive phase duration for
c      a specified burst as a function of radius
c
c      independent variables:
c       tscal = time scale factor
c       rslkt = 1-kiloton scale radius
c
c      dependent variable:
c       dtpp = positive phase duration
c
c      local variable:
c       dtlkt = 1-kiloton scale positive phase duration
c
      common/conp/ a0p,b0p2,c0p,a1p,b1p,r0p
c
c      apply the lower limit to the radius
      r1=amax1(rslkt,r0p)
c
c      compute the 1-kt positive phase duration
      dtlkt=c0p*(1.0-a1p/r1**b1p)*sqrt(alog(a0p*sqrt(r1*r1+b0p2)))
c
c      scale the positive phase duration to actual
      dtpp=dtlkt/tscal
c
      return
c
      end
c-----------------------------------------------------------------------
```

Figure A-4. Functions *dist, tshk,* and *dtpp* from the waveform module.

```
c______________________________________________________________________
c______________________________________________________________________
      function peak(pscal,rs1lb)
c
c     computes the peak overpressure as a function
c     of scaled radius using the kingery fit
c
c      independent variables:
c       pscal = pressure scale factor
c       rs1lb = 1-pound scale radius
c
c      dependent variable:
c       peak = peak overpressure
c
      common/conv/ okt,ocm,odsc,ocmkt3
      common/conk/ ak0,ak(8)
c
c      compute the peak overpressure in psi
      alrs1=alog(rs1lb)
      ping=ak0
      do 110 i=1,8
        ping=ping+ak(i)*alrs1**i
  110 continue
      ping=exp(ping)
c
c      convert to dyne/cm**2 and scale to sea level
      peak=odsc*pscal*ping
c
      return
c
      end
c______________________________________________________________________
c______________________________________________________________________
c______________________________________________________________________
      function reak(prtio,rambi)
c
c      computes peak density (adapted from LAMB)
c
c      independent variables:
c       prtio = pressure ratio
c       rambi = ambient density
c
c      dependent variable
c       reak = peak density:
c
      common/coni/ gamma,gmone,gpone
c
      reak=rambi*((gpone*prtio+gmone)/(gmone*prtio+gpone))
c
      return
c
      end
c______________________________________________________________________
c______________________________________________________________________
c______________________________________________________________________
      function veak(ppeak,rpeak,rambi)
c
c      computes the peak particle velocity (adapted from LAMB)
c
c      independent variables:
c       rambi = ambient density
c       ppeak = peak overpressure
c       rpeak = peak overdensity
c
c      dependent variable:
c       veak = peak particle velocity
c
      opeak=rpeak-rambi
      veak=sqrt(ppeak*opeak/(rambi*(rambi+opeak)))
c
      return
c
      end
c______________________________________________________________________
```

Figure A-5. Functions *peak, reak,* and *veak* from the waveform module.

Function *reak*, adapted from LAMB, is shown in Figure A-5. It computes the peak density of the blast wave associated with a specified burst given the pressure ratio and the ambient density. It is only used in conjunction with the LAMB peak overpressure combination algorithm.

Function *veak*, adapted from LAMB, is shown in Figure A-5. It computes the peak particle velocity of the blast wave associated with a specified burst given the peak overpressure and overdensity, and the ambient density. It is only used in conjunction with the LAMB peak overpressure combination algorithm.

COALESCENCE MODULE

These routines determine the effective yield at each station for a specified initial donor stack. It is recognized that more than one group of coalesced waves may pass a station and the yield selected is that associated with the group that produces the largest combined peak overpressure.

Subroutine *wave* is shown in Figure A-6. It cycles through the stacks, computes the required scale factors with calls to *sckt*, computes the stack-to-station radii with calls to *dist*, and computes the waveform variables with calls to *tshk*, *dtpp*, and *peak*, as well as to *reak* and *veak* when the LAMB model is specified.

Subroutine *ssup* is shown in Figure A-7. It totals the yields and, using simple superposition, combines the peak overpressures associated with coalesced blast waves. This option can only be selected by altering the default selection in the file bwaco.cfg.

Subroutine *lamb* is shown in Figure A-7. It totals the yields and, using the LAMB algorithm, combines the peak overpressures associated with coalesced blast waves. This option can only be selected by altering the default selection in the file bwaco.cfg.

Subroutine *comb* is shown in Figure A-7. It totals the yields and, using the total yield, computes the peak overpressures associated with coalesced blast waves.

Subroutine *coal* is shown in Figure A-8. It is given the index of a specified stack and the coordinates of the current station. It cycles through all the other stacks and, using the shock and zero-overpressure arrival times for all stacks at the current station, it applies the coalescence criterion that the shock from

```
c
c
      subroutine wave(istan,jstan)
c
c      cycles through the stacks and computes the waveform variables
c
c      independent variables:
c        istan = station x-index
c        jstan = station y-index
c
c      configuration variable:
c        icomb = pressure combination algorithm switch
c
c      station variables:
c        xstat = array of x-coordinates for the stations
c        ystat = array of y-coordinates for the stations
c        zstat = z-coordinate of stations
c
c      stack variables:
c        nstak = total number of stacks
c        xstak = array of x-coordinates for the stacks
c        ystak = array of y-coordinates for the stacks
c        zstak = z-coordinate for the stacks
c        wstak = array of yields for the stacks
c        tstak = array of stack initiation times
c        tshok = array of shock arrival times for waves from stacks
c        tzero = array of zero-overpressure arrival times
c                for waves from stacks
c        ppeak = array of peak overpressures for waves from stacks
c        rpeak = array of peak densities for waves from stacks
c        vpeak = array of peak velocities for waves from stacks
c
c      local variables:
c        istak = stack counter
c        tscal = time scale factor
c        rscal = radius scale factor
c        pscal = pressure scale factor
c        rstst = stack to station radius
c        rslkt = radius scaled to 1 kiloton
c        rsllb = radius scaled to 1 pound
c        prtio = pressure ratio
c
      common/conv/ okt,ocm,odsc,ocmkt3
      common/icfg/ icomb,iscrn,iprnt
      common/ambi/ pambi,rambi
      common/sttp/ xstat(120),ystat(120),zstat
      common/stkn/ nstak
      common/stkp/ xstak(100),ystak(100),zstak
      common/stkt/ vstak(100),tstak(100)
      common/stky/ wstak(100)
      common/stkw/ tshok(100),tzero(100),
     +             ppeak(100),rpeak(100),vpeak(100)
c
c      cycle through the stacks
      do 110 istak=1,nstak
c
c        compute the scale factors
        call sckt(zstak,wstak(istak),zstat, tscal,rscal,pscal)
c
c        compute the stack to station radius
        rstst=dist(xstat(istan),ystat(jstan),zstat,
     +             xstak(istak),ystak(istak),zstak)
c
c        scale the radius to 1-kt
        rslkt=rstst/rscal
c
c        compute the shock and zero-overpressure arrival times
        tshok(istak)=tshk(tscal,rslkt)+tstak(istak)
        tzero(istak)=dtpp(tscal,rslkt)+tshok(istak)
c
c        scale the radius to 1-lb and nondoubled yield
        rsllb=ocmkt3*rslkt
c
c        compute the peak overpressure, density and particle velocity
        ppeak(istak)=peak(pscal,rsllb)
        if(icomb.eq.1) then
          prtio=(ppeak(istak)+pambi)/pambi
          rpeak(istak)=reak(prtio,rambi)
          vpeak(istak)=veak(ppeak(istak),rpeak(istak),rambi)
          rpeak(istak)=rpeak(istak)-rambi
        endif
c
  110 continue
c
      return
c
      end
c
```

Figure A-6. Subroutine *wave* from the coalescence module.

```
c
c
      subroutine ssup(ncoal,wcoal,pcoal, wcomb,pcomb)
c
c      combines yields and peak overpressures using simple superposition
c
c      independent variables:
c       ncoal = number of coalesced waves
c       wcoal = array of yields of sources of coalesced waves
c       pcoal = array of peak overpressures from coalesced waves
c
c      dependent variables:
c       wcomb = combined yield from sources of all coalesced waves
c       pcomb = combined peak overpressure of all coalesced waves
c
c      local variable:
c       iwave = wave counter
c
      dimension wcoal(100),pcoal(100)
c
      wcomb=0.
      pcomb=0.
      do 110 iwave=1,ncoal
        wcomb=wcomb+wcoal(iwave)
        pcomb=pcomb+pcoal(iwave)
  110 continue
c
      return
c
      end
c
c
c
       subroutine lamb(ncoal,rasta,wcoal,pcoal,rcoal,vcoal, wcomb,pcomb)
c
c      combines yields and peak overpressures using LAMB algorithm
c
c      independent variables:
c       ncoal = number of coalesced waves
c       rasta = ambient density at the current station
c       wcoal = array of yields of sources of coalesced waves
c       pcoal = array of peak overpressures from coalesced waves
c       rcoal = array of peak densities from coalesced waves
c       vcoal = array of peak particle velocities from coalesced waves
c
c      dependent variables:
c       wcomb = combined yield from sources of all coalesced waves
c       pcomb = combined peak overpressure of all coalesced waves
c
c      local variables:
c       iwave = wave counter
c       rcomb = combined peak density of all coalesced waves
c       vcomb = combined peak velocity of all coalesced waves
c       ecomb = combined peak kinetic energy of all coalesced waves
c
      dimension wcoal(100),pcoal(100),rcoal(100),vcoal(100)
c
      wcomb=0.
      rcomb=rasta
      pcomb=0.
      vcomb=0.
      ecomb=0.
      do 110 iwave=1,ncoal
        wcomb=wcomb+wcoal(iwave)
        rcomb=rcomb+rcoal(iwave)
        rv=(rasta+rcoal(iwave))*vcoal(iwave)
        vcomb=vcomb+rv
        ecomb=ecomb+rv*vcoal(iwave)
        pcomb=pcomb+pcoal(iwave)
  110 continue
      vcomb=vcomb/rcomb
      pcomb=pcomb+0.6*(ecomb-rcomb*vcomb*vcomb)
c
      return
c
      end
c
c
c
      subroutine comb(ncoal,xcoal,ycoal,wcoal,xstat,ystat,zstat,wcomb,
     +                pcomb)
c
c      combines peak overpressures using pressure from combined stacks
c
c      independent variables:
c       ncoal = number of coalesced waves
c       xcoal = array of x-coordinates of sources of coalesced waves
c       ycoal = array of y-coordinates of sources of coalesced waves
c       wcoal = array of yields of sources of coalesced waves
c       xstat = x-coordinate of the current station
c       ystat = y-coordinate of the current station
c       zstat = z-coordinate of the current station
c
c      dependent variables:
c       wcomb = combined yield from sources of all coalesced waves
c       pcomb = combined peak overpressure of all coalesced waves
c
c      stack variable:
c       zstak = z-coordinate for the stacks
c
c      local variables:
c       tcscl = time scale factor for combined stacks
c       rcscl = radius scale factor for combined stacks
c       pcscl = pressure scale factor for combined stacks
c       xcomb = x-coordinate of center of combined stacks
c       ycomb = y-coordinate of center of combined stacks
c       rsllb = 1-lb scale radius
c
      common/conv/ okt,ocm,odsc,ocmkt3
      common/stkp/ xstak(100),ystak(100),zstak
c
      dimension xcoal(100),ycoal(100),wcoal(100)
c
c      find weight and center of combined stacks
      call sseq(ncoal,xcoal,ycoal,wcoal, xcomb,ycomb,wcomb)
c
c      compute the scale factors
      call sckt(zstak,wcomb,zstat, tcscl,rcscl,pcscl)
c
c      compute the 1-lb scale radius from the center of
c      the combined stacks using the nondoubled yield
      rsllb=ocmkt3*dist(xstat,ystat,zstat,xcomb,ycomb,zstak)/rcscl
c
c      compute the combined pressure
      pcomb=peak(pcscl,rsllb)
c
c
      return
c
      end
c
```

Figure A-7. Subroutines *ssup*, *lamb*, and *comb* from the coalescence module.

```
c_____________________________________________________________________
c
      subroutine coal(ispec,xstat,ystat,zstat,rasta, ncoal,wcomb,pcomb)
c
c       applies the coalescence criterion and determines the
c       total yield and combined peak overpressure from all
c       waves which coalesce with the wave from a specified stack
c
c       independent variables:
c        ispec = index of the specified stack
c        xstat = x-coordinate of the current station
c        ystat = y-coordinate of the current station
c        zstat = z-coordinate of the current station
c        rasta = ambient density at the current station
c
c       dependent variables:
c        ncoal = number of coalesced waves
c        wcomb = combined yield of the sources of all colaesced waves
c        pcomb = combined peak overpressure of all coalesced waves
c
c       configuration variable:
c        icomb = pressure combination algorithm switch
c
c       stack variables:
c        nstak = total number of stacks
c        xstak = array of x-coordinates for the stacks
c        ystak = array of y-coordinates for the stacks
c        wstak = array of yields for the stacks
c        ppeak = array of peak overpressures for waves from the stacks
c        rpeak = array of peak densities for waves from the stacks
c        vpeak = array of peak velocities for waves from the stacks
c        tshok = array of shock arrival times for waves from the stacks
c        tzero = array of zero-overpressure arrival times
c                for waves from the stacks
c
c       local variables:
c        istak = stack counter
c        xcoal = array of x-coordinates of sources of coalesced waves
c        ycoal = array of y-coordinates of sources of coalesced waves
c        wcoal = array of yields of sources of coalesced waves
c        pcoal = array of peak overpressures from coalesced waves
c        rcoal = array of peak densities from coalesced waves
c        vcoal = array of peak particle velocities from coalesced waves
c
      common/icfg/ icomb,iscrn,iprnt
      common/stkn/ nstak
      common/stkp/ xstak(100),ystak(100),zstak
      common/stky/ wstak(100)
      common/stkw/ tshok(100),tzero(100),
     +             ppeak(100),rpeak(100),vpeak(100)
c
      dimension xcoal(100),ycoal(100),wcoal(100),
     +          pcoal(100),rcoal(100),vcoal(100)
c
c       initialize the number of coalesced waves and coalescence arrays
      ncoal=1
      xcoal(1)=xstak(ispec)
      ycoal(1)=ystak(ispec)
      wcoal(1)=wstak(ispec)
      pcoal(1)=ppeak(ispec)
      if(icomb.eq.1) then
        rcoal(1)=rpeak(ispec)
        vcoal(1)=vpeak(ispec)
      endif
c
c       cycle through the stacks
      do 110 istak=1,nstak
c
c         if the current stack is not the specified stack
        if(istak.ne.ispec) then
c
c           if the wave from the current stack is coalesced
c           with the wave from the specified stack
          if(tshok(istak).ge.tshok(ispec)) then
            if(tshok(istak).le.tzero(ispec)) then
c
c               increment the number of coalesced waves
              ncoal=ncoal+1
c
c               store the peak overpressure, overdensity and velocity
              xcoal(ncoal)=xstak(istak)
              ycoal(ncoal)=ystak(istak)
              wcoal(ncoal)=wstak(istak)
              pcoal(ncoal)=ppeak(istak)
              if(icomb.eq.1) then
                rcoal(ncoal)=rpeak(istak)
                vcoal(ncoal)=vpeak(istak)
              endif
c
            endif
          endif
c
        endif
c
  110 continue
c
c
c         combine the yields and peak
c         overpressures of the coalesced waves
      if(ncoal.gt.1) then
c
c         use simple superposition
        if(icomb.eq.0) call ssup(ncoal,wcoal,pcoal, wcomb,pcomb)
c
c         use the LAMB algorithm
        if(icomb.eq.1) call lamb(ncoal,rasta,wcoal,pcoal,rcoal,vcoal,
     +                           wcomb,pcomb)
c
c         use the combined stacks
        if(icomb.eq.2) call comb(ncoal,xcoal,ycoal,wcoal,xstat,ystat,
     +                           zstat, wcomb,pcomb)
c
      else
c
c         use the yield and peak overpressure from the specified stack
        wcomb=wcoal(1)
        pcomb=pcoal(1)
c
      endif
c
      return
c
      end
c_____________________________________________________________________
```

Figure A-8. Subroutine *coal* from the coalescence module.

one wave lie in the positive overpressure phase of another wave. In this way, it determines the number of stacks producing waves which coalesce with the wave from the specified stack as well as the total yield and combined peak overpressure (using calls to *ssup*, *lamb*, or *comb* depending on how the session is configured) of the coalesced waves. It returns the number of coalesced waves as well as the associated total yield and combined peak overpressure.

Subroutine *yeff* is shown in Figure A-9. It is given the coordinates of the current station. It cycles through the stacks, calling *coal*, and determines the effective yield associated with the group of coalesced waves producing the largest combined peak overpressure at the current station. It returns the number of coalesced waves in the group, the effective yield and the combined peak overpressure.

Subroutine *effy* is shown in Figure A-10. It cycles through the stations and computes the waveform variables for each stack with calls to *wave* and the effective yields and combined peak overpressures with calls to *yeff*. It saves the peak overpressure and effective yield associated with the largest combined peak overpressure at each station.

PROPAGATION MODULE

These routines determine the order in which the stacks detonate and the time of detonation of each stack for a given initial donor.

Function *tdly* is shown in Figure A-11. It is given the coordinates of a donor and an acceptor stack as well as the propagation velocity of the donor. It computes and returns the initiation delay (communication time) between the donor and acceptor stack.

Subroutine *arly* is shown in Figure A-12. It is given the coordinates, initiation time, and propagation velocity of a donor stack. It cycles through all the other stacks which have not yet become donors and, with calls to *tdly*, determines the initiation time of the earliest acceptor for that donor. It assigns that time to the stack initiation time array if it is earlier than the time already assigned. It returns the index of that acceptor and its initiation time.

```
c______________________________________________________________________
c
      subroutine yeff(xstat,ystat,zstat,rambi, icsta,wstat,pstat)
c
c         computes the effective yield and combined
c         peak overpressure at a current station
c
c       independent variables:
c        xstat = x-coordinate of the current station
c        ystat = y-coordinate of the current station
c        zstat = z-coordinate of the current station
c        rambi = ambient density at the current station
c
c       dependent variables:
c        icsta = coalescence status at the current station
c        wstat = effective yield at the current station
c        pstat = combined peak overpressure at the current station
c
c       stack variable:
c        nstak = total number of stacks
c
c       local variables:
c        istak = stack counter
c        ncoal = number of waves coalesced with
c                the wave from the current stack
c        wcomb = combined yield at the current station
c        pcomb = combined peak overpressure at the current station
c
      common/stkn/ nstak
c
c       initialize the coalescence status, effective
c       yield and combined peak overpressure
      icsta=0
      wstat=0.
      pstat=0.
c
c       cycle through the stacks
      do 110 istak=1,nstak
c
c         determine the total yield and combined peak overpressure from
c         all waves which coalesce with the wave from the current stack
         call coal(istak,xstat,ystat,zstat,rambi, ncoal,wcomb,pcomb)
c
c          save the coalescence status, total yield
c          and combined peak overpressure associated
c          with the largest combined peak overpressure
         if(pcomb.gt.pstat) then
            icsta=ncoal
            wstat=wcomb
            pstat=pcomb
         endif
c
  110 continue
c
      return
c
      end
c______________________________________________________________________
```

Figure A-9. Subroutine *yeff* from the coalescence module.

```
c
c
      subroutine effy
c
c       computes the effective yield associated with the
c       largest combined peak overpressure at each station
c
c       station variables:
c        nstat = number of stations
c        lblok = length of station blocks
c        nblok = number of station blocks
c        xstat = array of x-coordinates for the stations
c        ystat = array of y-coordinates for the stations
c        zstat = z-coordinate of stations
c        icoal = array of coalescence statuses for the stations
c        wstat = array of effective yields for the stations
c        pstat = array of effective peak overpressures for the stations
c        rambi = ambient density
c
c       local variables:
c        istat = station counter
c        istan = station x-index
c        jstan = station y-index
c        iccur = coalescence status at current station
c        wscur = effective yield at current station
c        pscur = combined peak overpressure at current station
c
      common/alow/ pallo
      common/ambi/ pambi,rambi
      common/sttn/ nstat
      common/sttb/ lblok,nblok
      common/sttp/ xstat(120),ystat(120),zstat
      common/sttv/ icoal(3600),pstat(3600),wstat(3600)
c
c      initialize the station report
      write(6,2010) 0,nstat
c
c      cycle through the stations
      do 110 istat=1,nstat
c
c         report the current station
        write(6,2020) istat,nstat
c
c         compute the station indices
        istan=mod(istat-1,lblok)+1
        jstan=(istat-1)/lblok+1
c
c         compute waveform variables
        call wave(istan,jstan)
c
c         compute the effective yield and combined
c         peak overpressure at the current station
        call yeff(xstat(istan),ystat(jstan),zstat,rambi,
     +            iccur,wscur,pscur)
c
c         save the effective yield and combined peak
c         overpressure associated with the largest
c         combined peak overpressure for any donor
        if(pscur.gt.pstat(istat)) then
          if(pscur.lt.pallo) then
            icoal(istat)=0
            wstat(istat)=0.
          else
            icoal(istat)=iccur
            wstat(istat)=wscur
          endif
          pstat(istat)=pscur
        endif
c
  110 continue
c
      return
c
c      format statements
 2010 format('    Computing coalescence at station ',i4,' of ',i4)
 2020 format('+   Computing coalescence at station ',i4,' of ',i4)
c
      end
c
```

Figure A-10. Subroutine *effy* from the coalescence module.

```
c_______________________________________________________________________
c
      function tdly(xacpt,yacpt,xdonr,ydonr,vdonr)
c
c      computes the initiation delay between a given donor and acceptor
c
c       independent variables:
c        xacpt = x-coordinate of the acceptor
c        yacpt = y-coordinate of the acceptor
c        xdonr = x-coordinate of the donor
c        ydonr = y-coordinate of the donor
c        vdonr = propagation velocity of the donor
c
c       dependent variable:
c        tdly = initiation delay
c
c       local variables:
c        deltx = x-coordinate difference
c        delty = y-coordinate difference
c
      if(vdonr.eq.0.0) then
c
c         infinite propagation velocity
        tdly=0.0
c
      else
c
c         finite propagation velocity
        deltx=xacpt-xdonr
        delty=yacpt-ydonr
        tdly=sqrt(deltx*deltx+delty*delty)/vdonr
c
      endif
c
      return
c
      end
c_______________________________________________________________________
```

Figure A-11. Function *tdly* from the propagation module.

```
c
c
      subroutine arly(xdonr,ydonr,tdonr,vdonr, ierly,terly)
c
c      computes earliest acceptor for a given donor
c
c      independent variables:
c       xdonr = x-coordinate of donor
c       ydonr = y-coordinate of donor
c       tdonr = initiation time of donor
c       vdonr = propagation velocity of donor
c
c      stack variables:
c       xstak = array of x-coordinates for the stacks
c       ystak = array of y-coordinates for the stacks
c       tstak = array of initiation times for the stacks
c       mstak = array of initiation statuses for the stacks
c
c      dependent variables:
c       ierly = index of earliest acceptor
c       terly = initiation time of earliest acceptor
c
c      local variable:
c       istak = stack counter
c
      common/stkn/ nstak
      common/stkp/ xstak(100),ystak(100),zstak
      common/stkt/ vstak(100),tstak(100)
      common/inst/ mstak(100)
c
c      initialize earliest initiation time
      terly=1.e9
c
c        cycle through the stacks
      do 110 istak=1,nstak
c
c         if the current stack is not already a donor
        if(mstak(istak).eq.0) then
c
c           compute its initiation time with respect to given donor
          tstak(istak)=tdonr+tdly(xstak(istak),ystak(istak),xdonr,ydonr,
     +                 vdonr)
c
c           save the earliest initiation time
          if(tstak(istak).lt.terly) then
            terly=tstak(istak)
            ierly=istak
          endif
c
        endif
c
  110 continue
c
      return
c
      end
c
```

Figure A-12. Subroutine *arly* from the propagation module.

Subroutine *prop* is shown in Figure A-13. It is given the index of the current initial donor stack. It cycles through all the stacks which have become donors to determine the next donor using calls to *arly*. It repeats the process using that donor and continues until initiation times have been assigned to all of the stacks.

EXECUTION MODULE

These routines read the user's input, set up the requested computation, and execute it.

Subroutine *inpt*, shown in Figure A-14, reads the user's input from the file *bwaco.inp*.

Subroutine *cent*, shown in Figure A-15, computes either coordinate of the center of charge of a given stack arrangement.

Subroutine *sseq*, shown in Figure A-15, calls *cent* to compute both coordinates of the center of charge of a given stack arrangement.

Subroutine *stak*, shown in Figure A-16, calls *sseq* to compute the single-stack equivalent of all the stacks, converts the stack variables to 1-KT Standard units and doubles their yields to simulate the effects of the ground plane.

Subroutine *blok*, shown in Figure A-17, determines the station arrangement that best covers the region of interest.

Subroutine *stns*, shown in Figure A-18, determines the region of interest (if none is specified by the user), determines the station arrangement by calling *blok*, defines the station coordinates, and initializes the station variables.

Subroutine *rtim*, shown in Figure A-19, computes and reports the run time for the coalescence computation.

```
c
c
      subroutine prop(idoni)
c
c      computes initiation times of stacks
c
c      independent variable:
c       idoni = index of initial donor
c
c      stack variables:
c       nstak = number of stacks
c       xstak = array of x-coordinates for the stacks
c       ystak = array of y-coordinates for the stacks
c       vstak = array of stack propagation velocities
c       tstak = array of initiation times for the stacks
c       mstak = array of initiation statuses for the stacks
c
c      local variables:
c       istak = stack counter
c       ierly = index of earliest acceptor
c       terly = initiation time of earliest acceptor
c       idonx = index of next donor
c       tnext = earliest acceptor for all donors
c       ndonr = number of identified donors
c
      common/stkn/ nstak
      common/stkp/ xstak(100),ystak(100),zstak
      common/stkt/ vstak(100),tstak(100)
      common/inst/ mstak(100)
c
c      initialize stack initiation statuses and times
      do 110 istak=1,nstak
        mstak(istak)=0
        tstak(istak)=1.e9
  110 continue
c
c      initialize donor initiation status and time
      ndonr=1
      mstak(idoni)=1
      tstak(idoni)=0.
c
c      find earliest acceptor for all donors
  120 tnext=1.e9
c
c      cycle through the stacks
      do 130 istak=1,nstak
c
c        if the current stack is an identified donor
        if(mstak(istak).gt.0) then
c
c          determine the earliest acceptor for that donor
          call arly(xstak(istak),ystak(istak),tstak(istak),vstak(istak),
     +              ierly,terly)
c
c          save the earliest acceptor for all donors
          if(terly.lt.tnext) then
            idonx=ierly
            tnext=terly
          endif
c
        endif
c
  130 continue
c
c      increment donor
      ndonr=ndonr+1
c
c      assign earliest acceptor as next donor
      mstak(idonx)=ndonr
      tstak(idonx)=tnext
c
c      check for completion of stack initiation time assignments
      if(ndonr.lt.nstak) go to 120
c
      return
c
      end
c
```

Figure A-13. Subroutine *prop* from the propagation module.

```
c-----------------------------------------------------------------------
c
      subroutine inpt
c
c      obtains problem input from the file bwaco.inp
c
c      problem variable:
c       ulabl = user's problem label
c
c      stack variables:
c       nstak = total number of stacks
c       xstak = array of x-coordinates for the stacks
c       ystak = array of y-coordinates for the stacks
c       zstak = z-coordinate for the stacks
c       wstak = array of yields for the stacks
c       vstak = array of stack propagation velocities
c       idonr = array of initial donor switches
c
c      region variables:
c       xmin = minimum x-coordinate of region of interest
c       xmax = maximum x-coordinate of region of interest
c       ymin = minimum y-coordinate of region of interest
c       ymax = maximum y-coordinate of region of interest
c
      common/ulab/ ulabl
      common/stkn/ nstak
      common/stkp/ xstak(100),ystak(100),zstak
      common/stkt/ vstak(100),tstak(100)
      common/stky/ wstak(100)
      common/stkd/ idonr(100),nidnr
      common/mxmn/ xmin,xmax,ymin,ymax
c
      character*80 ulabl,dummy
c
c      log call
      write(8,3010)
c
c      open the user input file
      open(7,file='bwaco.inp',form='formatted',status='old',err=140)
c
c      read the user's problem label
      read(7,1010,end=160,err=150) ulabl
      write(8,3020) ulabl
c
c      read the region data
      read(7,1010,end=160,err=150) dummy
      read(7,1020,end=160,err=150) xmin,xmax,ymin,ymax
      write(8,3030) xmin,xmax,ymin,ymax
c
c      read the stack data
      read(7,1010,end=160,err=150) dummy
      nstak=0
      zstak=0.
  110 nstak=nstak+1
        if(nstak.gt.100) go to 120
        read(7,1020,end=120,err=150)
     +  xstak(nstak),ystak(nstak),wstak(nstak),vstak(nstak),idonr(nstak)
        write(8,3040)
     +  xstak(nstak),ystak(nstak),wstak(nstak),vstak(nstak),idonr(nstak)
      go to 110
c
c      close the user input file and adjust the number of stacks
  120 close(7)
      nstak=nstak-1
c
c      remove any bogus stacks
  130 continue
      if(xstak(nstak).eq.0..and.ystak(nstak).eq.0..and.
     +   wstak(nstak).eq.0..and.vstak(nstak).eq.0..and.
     +   idonr(nstak).eq.0) then
        nstak=nstak-1
        go to 130
      endif
c
      if(nstak.lt.1) go to 170
c
      write(8,3050) nstak
c
      return
c
c      errors detected
  140 stop ' BWACO: ERROR - Unable to open bwaco.inp!'
  150 stop ' BWACO: ERROR - Unable to read bwaco.inp!'
  160 stop ' BWACO: ERROR - Premature end of bwaco.inp!'
  170 stop ' BWACO: ERROR - Fewer than one stack specefied!'
c
c      format statements
 1010 format(a80)
 1020 format(4f10.2,4x,i1)
 3010 format(/' inpt called')
 3020 format('    label: 'a80)
 3030 format('   region: '4f10.2)
 3040 format('    stack: '4f10.2,i2)
 3050 format('   stacks: 'i3)
c
      end
c-----------------------------------------------------------------------
```

Figure A-14. Subroutine *inpt* from the execution module.

```
c_____________________________________________________________________
c
      subroutine sseq(nstck,xstck,ystck,wstck, xsseq,ysseq,wsseq)
c
c      computes the single stack equivalent of the distributed stacks
c
c       independent variables:
c        nstck = number of stacks
c        xstck = array of x-coordinates for the stacks
c        ystck = array of y-coordinates for the stacks
c        wstck = array of yields for the stacks
c
c       dependent variables:
c        xsseq = x-coordinate of center of charge
c        ysseq = y-coordinate of center of charge
c        wsseq = single stack equivalent weight
c
c       local variable:
c        istck = stack counter
c
      dimension xstck(100),ystck(100),wstck(100)
c
c       compute the single stack equivalent weight
      wsseq=0.
      do 110 istck=1,nstck
        wsseq=wsseq+wstck(istck)
  110 continue
c
c       compute the center of charge
      xsseq=cent(nstck,xstck,wstck,wsseq)
      ysseq=cent(nstck,ystck,wstck,wsseq)
c
      return
c
      end
c_____________________________________________________________________
c_____________________________________________________________________
c
      function cent(nstck,rstck,wstck,wtotl)
c
c       computes one coordinate of the center of charge
c
c       independent variables:
c        nstck = number of stacks
c        rstck = array of coordinates for the stacks
c        wstck = array of yields for the stacks
c        wtotl = total yield of stacks
c
c       local variable:
c        istck = stack counter
c
      dimension rstck(100),wstck(100)
c
      rwstk=0.
      do 110 istck=1,nstck
        rwstk=rwstk+rstck(istck)*wstck(istck)
  110 continue
      cent=rwstk/wtotl
c
      return
c
      end
c_____________________________________________________________________
```

Figure A-15. Subroutine *sseq* and function *cent* from the execution module.

```
c
c
      subroutine stak
c
c      defines the stacks
c
c      stack variables:
c       nstak = total number of stacks
c       xstak = array of x-coordinates for the stacks
c       ystak = array of y-coordinates for the stacks
c       zstak = z-coordinate for the stacks
c       wstak = array of yields for the stacks
c       vstak = array of stack propagation velocities
c       nidnr = number of initial donors specified
c
c      region variables:
c       xcent = x-coordinate of center of charge
c       ycent = y-coordinate of center of charge
c       rinhb = inhabited building radius
c       xmini = minimum x-coordinate
c       xmaxi = maximum x-coordinate
c       ymini = minimum y-coordinate
c       ymaxi = maximum y-coordinate
c
c      local variables:
c       istak = stack counter
c       dregn = region margin
c       rregn = region radius
c
      common/conf/ thrd,tpi
      common/conv/ okt,ocm,odsc,ocmkt3
      common/regn/ xcent,ycent,rinhb,xmini,xmaxi,ymini,ymaxi
      common/alow/ pallo
      common/stkn/ nstak
      common/stkp/ xstak(100),ystak(100),zstak
      common/stky/ wstak(100)
      common/stkt/ vstak(100),tstak(100)
      common/stkd/ idonr(100),nidnr
      common/totl/ wtotl
c
c      log call
      write(8,3010)
c
c      compute single charge equivalent and inhabited building radius
      call sseq(nstak,xstak,ystak,wstak, xcent,ycent,wtotl)
      if(wtotl.lt.3.0e4) then
        if(wtotl.gt.100.) then
          rinhb=380.
        else
          rinhb=204.
        endif
        pallo=8.3e4
      endif
      if(wtotl.ge.3.0e4.and.wtotl.lt.1.0e5) then
        rinhb=12.2*wtotl**thrd
        pallo=8.3e4
      endif
      if(wtotl.ge.1.0e5.and.wtotl.lt.2.5e5) then
        rinhb=0.738*wtotl**0.577
        pallo=9.7e4-0.14*wtotl
      endif
      if(wtotl.ge.2.5e5) then
        rinhb=15.24*wtotl**thrd
        pallo=6.2e4
      endif
      dregn=0.05*rinhb
      rregn=rinhb+dregn
      xmini=xcent-rregn
      xmaxi=xcent+rregn
      ymini=ycent-rregn
      ymaxi=ycent+rregn
      write(8,3020) rinhb,xmini,xmaxi,ymini,ymaxi
c
c      adjust the region limits, convert the stack variables to 1-kt
c      standard units and double the yields for ground plane simulation
      nidnr=0
      do 110 istak=1,nstak
        rstak=15.24*wstak(istak)**thrd+dregn
        xmini=amin1(xstak(istak)-rstak,xmini)
        xmaxi=amax1(xstak(istak)+rstak,xmaxi)
        ymini=amin1(ystak(istak)-rstak,ymini)
        ymaxi=amax1(ystak(istak)+rstak,ymaxi)
        xstak(istak)=xstak(istak)/ocm
        ystak(istak)=ystak(istak)/ocm
        wstak(istak)=2.*wstak(istak)/okt
        vstak(istak)=vstak(istak)/ocm
        write(8,3030)
     +   xstak(istak),ystak(istak),wstak(istak),vstak(istak)
        if(idonr(istak).ne.0) nidnr=nidnr+1
  110 continue
c
c      check the number of initial donors
      if(nidnr.eq.0)
     +  stop ' BWACO: ERROR - No initial donors were specified!'
      if(nidnr.lt.nstak) write(6,2010)
      write(8,3040) nidnr
c
      return
c
c      format statements
 2010 format(//' BWACO: WARNING - Some stacks are not initial donors!')
 3010 format(/' stak called')
 3020 format('   region: ',1p5e10.3)
 3030 format('    stack: ',4f10.2)
 3040 format('   donors: ',i3)
c
      end
c
```

Figure A-16. <u>Subroutine *stak* from the execution module.</u>

```
c-----------------------------------------------------------------------
c
      subroutine blok(lbmax,nstat,argiv, lblok,nblok)
c
c      determines the integer block length and number
c      of blocks that yields the block aspect ratio
c      that most closely matches a given aspect ratio
c
c      independent variables:
c       lbmax = maximum block length
c       nstat = number of stations
c       argiv = given aspect ratio
c
c      dependent variables:
c       lblok = length of blocks
c       nblok = number of blocks
c
c      determine the minimum block length
      lbmin=nstat/lbmax
c
c      initialize the counter
      numbr=0
c
c      cycle through all possible block lengths
      do 110 lblck=lbmin,lbmax
c
c        if there are an integral number of blocks of length lblck
        if(mod(nstat,lblck).eq.0) then
c
c          determine the number of blocks
          nblck=nstat/lblck
c
c          determine the block aspect ratio
          arblk=float(lblck)/float(nblck)
c
c          as soon as the block aspect ratio
c          exceeds the given aspect ratio
          if(arblk.ge.argiv) then
c
c            if a previous solution has been determined
            if(numbr.gt.0) then
c
c              compute the differences between the given aspect
c              ratio and the previous and current aspect ratios
              dold=abs(argiv-arblo)
              dcur=abs(argiv-arblk)
c
c              if the given aspect ratio is
c              closer to the previous aspect ratio
              if(dold.lt.dcur) then
c
c                return the block length corresponding
c                to the previous block aspect ratio
                lblok=lblko
                nblok=nblko
c
c              otherwise
              else
c
c                return the block length corresponding
c                to the curent block aspect ratio
                lblok=lblck
                nblok=nblck
c
              endif
c
c            otherwise
            else
c
c              return the block length corresponding
c              to the curent block aspect ratio
              lblok=lblck
              nblok=nblck
c
            endif
c
            return
c
          endif
c
c          increment the counter and save the current values
          numbr=numbr+1
          lblko=lblck
          nblko=nblck
          arblo=arblk
c
        endif
c
  110 continue
c
c      if a previous solution exists
      if(numbr.gt.0) then
c
c        return the block length corresponding
c        to the previous block aspect ratio
        lblok=lblko
        nblok=nblko
        return
c
c      otherwise
      else
c
c        an error condition exists
        stop ' BWACO: ERROR - Region cannot be blocked!'
c
      endif
c
      end
c-----------------------------------------------------------------------
```

Figure A-17. Subroutine *blok* from the execution module.

```
c
c
      subroutine stns
c
c      defines the stations
c
c      station variables:
c       nstat = number of stations
c       lblok = length of station bloks
c       nblok = number of station bloks
c       xstat = array of x-coordinates for the stations
c       ystat = array of y-coordinates for the stations
c       zstat = z-coordinate of stations
c       icoal = array of coalescence statuses for the stations
c       wstat = array of effective yields for the stations
c       pstat = array of effective peak overpressures for the stations
c
c      region variables:
c       rinhb = inhabited building radius
c       xmini = minimum x-coordinate
c       xmaxi = maximum x-coordinate
c       ymini = minimum y-coordinate
c       ymaxi = maximum y-coordinate
c
c      local variables:
c       istan = station index
c       jstan = station index
c       istat = station counter
c       xregn = x-dimension of region
c       yregn = y-dimension of region
c       asrat = aspect ratio of region
c       deltx = x-increment
c       delty = y-increment
c
      common/conv/ okt,ocm,odsc,ocmkt3
      common/icfg/ icomb,iscrn,iprnt
      common/mxmn/ xmin,xmax,ymin,ymax
      common/regn/ xcent,ycent,rinhb,xmini,xmaxi,ymini,ymaxi
      common/ambi/ pambi,rambi
      common/sttn/ nstat
      common/sttb/ lblok,nblok
      common/sttp/ xstat(120),ystat(120),zstat
      common/sttv/ icoal(3600),pstat(3600),wstat(3600)
c
c      log call
      write(8,3010)
c
c      check the region limits
      if(xmin.eq.xmax) then
        xmin=xmini
        xmax=xmaxi
      endif
      if(ymin.eq.ymax) then
        ymin=ymini
        ymax=ymaxi
      endif
      write(8,3020) xmin,xmax,ymin,ymax
c
c      convert the region limits to 1-kt standard units
      xmin=xmin/ocm
      xmax=xmax/ocm
      ymin=ymin/ocm
      ymax=ymax/ocm
      write(8,3020) xmin,xmax,ymin,ymax
c
c      define the number of stations
      nstat=3600
      lbmax=120
c
c      compute the coordinate ranges for the region
      xregn=xmax-xmin
      yregn=ymax-ymin
c
c      distribute the station indices
      asrat=xregn/yregn
      call blok(lbmax,nstat,asrat, lblok,nblok)
      write(8,3030) lblok,nblok
c
c      define the x values
      deltx=xregn/float(lblok)
      do 110 istan=1,lblok
        xstat(istan)=(float(istan-1)+0.5)*deltx+xmin
  110 continue
c
c      define the y values
      delty=yregn/float(nblok)
      do 120 jstan=1,nblok
        ystat(jstan)=(float(jstan-1)+0.5)*delty+ymin
  120 continue
c
      zstat=0.
c
c      initialize the coalescence status, effective yield,
c      overpressure and ambient pressure and density
      do 130 istat=1,nstat
        icoal(istat)=0
        wstat(istat)=0.
        pstat(istat)=0.
  130 continue
      call at62(0., pambi,dummy,rambi,dummy)
c
      return
c
c      format statement
 3010 format(/' stns called')
 3020 format('   region: ',4f10.2)
 3030 format('   blocks:',2i4)
c
      end
c
```

Figure A-18. Subroutine *stns* from the execution module.

```
c_______________________________________________________________________
c
      subroutine rtim(ihbgn,imbgn,isbgn,ilbgn,ihend,imend,isend,ilend)
c
c      determines and reports computation time
c
c      independent variables:
c       ihbgn = hour at which computation begins
c       imbgn = minute at which computation begins
c       isbgn = second at which computation begins
c       ilbgn = tenth second at which computation begins
c       ihend = hour at which computation ends
c       imend = minute at which computation ends
c       isend = second at which computation ends
c       ilend = tenth second at which computation ends
c
c      local variables:
c       tsbgn = time at which computation begins
c       tsend = time at which computation ends
c       tscmp = computation time
c
c      compute the computation time
      if(ihend.lt.ihbgn) ihend=ihend+24
      tsbgn=3600.*float(ihbgn)+60.*float(imbgn)
     +     +(float(isbgn)+float(ilbgn)/100.)
      tsend=3600.*float(ihend)+60.*float(imend)
     +     +(float(isend)+float(ilend)/100.)
      tscmp=tsend-tsbgn
c
c      report the computation time
      if(tscmp.lt.3600.) then
        if(tscmp.lt.60.) then
          write(6,2010) tscmp
        else
          write(6,2020) tscmp/60.
        endif
      else
        write(6,2030) tscmp/3600.
      endif
c
      return
c
c      format statements
 2010 format(/' Computation Time =',f7.2,' seconds'/)
 2020 format(/' Computation Time =',f7.2,' minutes'/)
 2030 format(/' Computation Time =',f7.2,' hours'/)
c
      end
c_______________________________________________________________________
```

Figure A-19. Subroutine *rtim* from the execution module.

Subroutine *copt* is shown in Figure A-20. It cycles through all possible donors (as specified by user input) computing the initiation times with calls to prop and the effective yields with calls to *effy*. It also obtains the beginning and ending times of the computations and calls *rtim* to compute and report the run time.

PLOTTING MODULE

These routines plot the results of the computations on the request of the user using proprietary graphics routines.

Subroutine *rest*, shown in Figure A-21, restores the original units and eliminates the ground plane factor.

Subroutine *spmn*, shown in Figure A-22, displays the BWACO plotting menu and obtains the user's request.

Subroutine *plts*, which is not listed, generates the coalescence and peak overpressure maps using calls to proprietary routines.

INTERFACE MODULE

These routines set up the session and direct the main BWACO functions at the discretion of the user.

Subroutine *smmn*, shown in Figure A-23, displays the BWACO main menu and obtains the user's request.

Subroutine *exec*, shown in Figure A-27, calls the routines of the execution module when the user requests that a problem be run.

Subroutine *rcfg*, shown in Figure A-24, reconfigures the session, allowing the user to select the default display and printer.

```
c
c
      subroutine copt
c
c      determines the worst-case coalescence pattern
c
c      stack variables:
c       nstak = total number of stacks
c       idonr = array of initial donor switches
c
c      local variables:
c       istak = stack counter
c       ihbgn = hour at which computation begins
c       imbgn = minute at which computation begins
c       isbgn = second at which computation begins
c       ilbgn = tenth second at which computation begins
c       ihend = hour at which computation ends
c       imend = minute at which computation ends
c       isend = second at which computation ends
c       ilend = tenth second at which computation ends
c
      common/stkn/ nstak
      common/stkd/ idonr(100),nidnr
c
c      log call
      write(8,3010)
c
c      obtain starting time
      call gettim(ihbgn,imbgn,isbgn,ilbgn)
c
c      cycle through the stacks
      idcnt=0
      do 110 istak=1,nstak
c
c        if the stack is a possible initial donor
         if(idonr(istak).eq.1) then
c
c          report the current initial donor number
           idcnt=idcnt+1
           write(6,2010) idcnt,nidnr
c
c          determine the propagation sequence
           call prop(istak)
c
c        determine the (worst-case) effective yields
           call effy
c
         endif
c
  110 continue
c
c      obtain completion time
      call gettim(ihend,imend,isend,ilend)
c
c      determine and report computation time
      call rtim(ihbgn,imbgn,isbgn,ilbgn,ihend,imend,isend,ilend)
c
      return
c
c      format statements
 2010 format( /' Initial Donor ',i3,' of ',i3,':')
 3010 format(/' copt called')
c
      end
c
```

Figure A-20. Subroutine *copt* from the execution module.

```
c
c
      subroutine rest
c
c      restores original units and plots the results
c
c      stack variables:
c       nstak = number of stacks
c       xstak = array of x-coordinates for the stacks
c       ystak = array of y-coordinates for the stacks
c
c      station variables:
c       lblok = length of station blocks
c       nblok = number of station blocks
c       nstat = number of stations
c       wstat = array of effective yields for the stations
c
c      local variables:
c       istak = stack counter
c       istan = station index
c       jstan = station index
c       istat = station counter
c
      common/conv/ okt,ocm,odsc,ocmkt3
      common/stkn/ nstak
      common/stkp/ xstak(100),ystak(100),zstak
      common/sttb/ lblok,nblok
      common/sttn/ nstat
      common/sttp/ xstat(120),ystat(120),zstat
      common/sttv/ icoal(3600),pstat(3600),wstat(3600)
c
c      log call
      write(8,3010)
c
c      restore the original units to the stack coordinates
      do 110 istak=1,nstak
        xstak(istak)=ocm*xstak(istak)
        ystak(istak)=ocm*ystak(istak)
  110 continue
c
c      restore the original units to the station coordinates
      do 120 istan=1,lblok
        xstat(istan)=ocm*xstat(istan)
  120 continue
      do 130 jstan=1,nblok
        ystat(jstan)=ocm*ystat(jstan)
  130 continue
c
c      restore the original units to and eliminate the
c      ground plane factor from the effective yields
      do 140 istat=1,nstat
        wstat(istat)=0.5*okt*wstat(istat)
  140 continue
c
      return
c
c      format statement
 3010 format(/' rest called')
c
      end
c
```

Figure A-21. Subroutine *rest* from the plotting module.

```
c
c
      subroutine spmn(irqst,iptyp,idevi)
c
c      displays plotting menu and obtains user request
c
c      dependent variables:
c       irqst = user request index
c       iptyp = index of requested plot type
c       idevi = index of requested device
c
c      log call
      write(8,3010)
c
c      display menu
      write(6,2010)
  110 write(6,2020)
      write(6,2030)
      write(6,2040)
      write(6,2050)
      write(6,2060)
      write(6,2070)
      write(6,2080)
c
c      obtain user request
      read(5,1010,err=110) irqst
      if(irqst.lt.0.or.irqst.gt.5) go to 110
      write(8,3020) irqst
c
c      assign requested plot type
      if(irqst.eq.1.or.irqst.eq.2) then
        iptyp=2
      else
        iptyp=1
      endif
c
c      assign requested device
      if(irqst.eq.1.or.irqst.eq.3) then
        idevi=iscrn
      else
        idevi=iprnt
      endif
c
      return
c
c      format statements
 1010 format(i1)
 2010 format(//' BWACO PLOTTING MENU')
 2020 format( /' Enter 1 to plot Coalescence Map (Screen)')
 2030 format(  '       2 to plot Coalescence Map (Printer)')
 2040 format(  '       3 to plot Peak Overpressure Map (Screen)')
 2050 format(  '       4 to plot Peak Overpressure Map (Printer)')
 2060 format(  '       5 to reconfigure the session')
 2070 format(  '       0 to return to the Main Menu')
 2080 format(  '       '\)
 3010 format(/' spmn called')
 3020 format('   irqst=' i1)
c
      end
c
```

Figure A-22. Subroutine *spmn* from the plotting module.

```
c
c
      subroutine smmn(iscur,irqst)
c
c      displays main menu and obtains user request
c
c      display menu heading
      write(6,2010)
c
c      check for presence of bwaco.dat
      open(7,file='bwaco.dat',status='old',err=120)
      close(7)
c
c      display complete menu
  110 write(6,2020)
      if(iscur.eq.0) then
        write(6,2030)
      else
        write(6,2031)
      endif
      write(6,2041)
      write(6,2050)
      write(6,2060)
c
c      obtain user request
      read(5,1010,err=110) irqst
c
c      check for valid request
      if(irqst.lt.0.or.irqst.gt.3) go to 110
c
      go to 130
c
c      display partial menu
  120 write(6,2020)
      write(6,2042)
      write(6,2050)
      write(6,2060)
c
c      obtain user request
      read(5,1010,err=120) irqst
c
c      check for valid request
      if(irqst.lt.0.or.irqst.gt.2) go to 120
c
c      adjust request number
      if(irqst.eq.2) irqst=3
c
 130  return
c
c      format statements
 1010 format(i1)
 2010 format(//' BWACO MAIN MENU ')
 2020 format(/' Enter 1 to run the current problem')
 2030 format( '       2 to plot the previous problem')
 2031 format( '       2 to plot the current problem')
 2041 format( '       3 to reconfigure the session')
 2042 format( '       2 to reconfigure the session')
 2050 format( '       0 to end the session')
 2060 format( '       '\)
c
      end
c
```

Figure A-23. Subroutine *smmn* from the interface module.

```
c
c---------------------------------------------------------------------
      subroutine rcfg(icfsm,ifrom)
c
c      reconfigures the session and rewrites the bwaco.cfg file
c
c       independent variables:
c        icfsm - mandatory configuration save switch
c        ifrom - calling routine index
c
c       configuration variables:
c        icomb - pressure combination algorithm switch
c        iscrn - default display switch
c        iprnt - default printer switch
c
      common/icfg/ icomb,iscrn,iprnt
c
      character*24 lable,lablp
c
c      log call
      write(8,3010)
c
c
c      assign the current default display to the menu heading
  110 if(iscrn.eq. 1) lable='CGA Mode  4 320x200/4'
      if(iscrn.eq. 2) lable='CGA Mode  5 320x200/4'
      if(iscrn.eq. 3) lable='CGA Mode  6 640x200/BW'
      if(iscrn.eq. 4) lable='Hercules     720x348/BW'
      if(iscrn.eq. 5) lable='EGA Mode 13 320x200/16'
      if(iscrn.eq. 6) lable='EGA Mode 14 640x200/16'
      if(iscrn.eq. 7) lable='EGA Mode 15 640x350/BW'
      if(iscrn.eq. 8) lable='EGA Mode 16 640x350/16'
      if(iscrn.eq. 9) lable='VGA Mode 17 640x480/BW'
      if(iscrn.eq.10) lable='VGA Mode 18 640x480/16'
      if(iscrn.eq.11) lable='ATI Mode 81 640x480/16'
      if(iscrn.eq.12) lable='ATI Mode 82 752x410/16'
      if(iscrn.eq.13) lable='ATI Mode 83 800x560/16'
c
c      assign the current default printer to the menu heading
      if(iprnt.eq.14) lablp='Single-density printer'
      if(iprnt.eq.15) lablp='Double-density printer'
      if(iprnt.eq.16) lablp='Quad-density printer'
      if(iprnt.eq.17) lablp='Epson Print  plot mode'
      if(iprnt.eq.18) lablp='HP Plotter (Scaling)'
      if(iprnt.eq.19) lablp='HP Plotter (Absolute)'
      if(iprnt.eq.20) lablp='HP Laser Jet (75 dpi)'
      if(iprnt.eq.21) lablp='HP Laser Jet (100 dpi)'
      if(iprnt.eq.22) lablp='HP Laser Jet (150 dpi)'
      if(iprnt.eq.23) lablp='HP Laser Jet (300 dpi)'
      if(iprnt.eq.24) lablp='HP Think Jet  Lo Dens'
      if(iprnt.eq.25) lablp='HP Think Jet  Hi Dens'
      if(iprnt.eq.26) lablp='HP Paint Jet  Lo Dens'
      if(iprnt.eq.27) lablp='HP Paint Jet  Hi Dens'
      if(iprnt.eq.28) lablp='Postscript Printer'
      if(iprnt.eq.29) lablp='Encapsulated Postscript'
c
c      display the current selections
      write(6,2010)
      write(6,2020) lable
      write(6,2030) lablp
c
c      display the configuration menu
  120 write(6,2040)
      write(6,2050)
      if(icfsm.eq.0) write(6,2060)
      if(ifrom.eq.1) write(6,2071)
      if(ifrom.eq.2) write(6,2072)
      if(ifrom.eq.3) write(6,2073)
      write(6,2080)
c
c      obtain the user request
      read(5,1010,err=120) ichng
c
c      check the validity of the request
      if(ichng.lt.0.or.ichng.gt.(3-icfsm)) go to 120
c
      if(ichng.eq.1) then
c
c        process a request to change the default display
        write(6,2110)
        write(6,2120)
        write(6,2130)
        write(6,2140)
        write(6,2150)
        write(6,2160)
        write(6,2170)
  130   write(6,2180)
c
c        obtain the requested display
        read(5,1020,err=130) iscrn
c
c        check the validity of the request
        if(iscrn.lt.1.or.iscrn.gt.13) go to 130
c
c        log the requested display
        write(8,3020) iscrn
c
      endif
c
      if(ichng.eq.2) then
c
c        process a request to change the default printer
        write(6,2210)
        write(6,2220)
c       write(6,2230)
        write(6,2240)
        write(6,2250)
        write(6,2260)
        write(6,2270)
        write(6,2280)
  140   write(6,2290)
c
c        obtain the requested printer
        read(5,1020,err=140) iprnt
c
c        check the validity of the request
        if(iprnt.lt.14.or.iprnt.gt.29) go to 140
        if(iprnt.eq.18.or.iprnt.eq.19) go to 140
c
c        log the requested printer
        write(8,3030) iprnt
c
      endif
c
      if(icfsm.eq.1.or.ichng.eq.3) then
c
c        process a request to save the configuration
        open(7,file='bwaco.cfg',form='formatted',err=170)
        write(7,1030) icomb,iscrn,iprnt
        close(7)
c
c        log save
        write(8,3040)
c
      endif
c
      if(ichng.ne.0) go to 110
c
      return
c
c      error detected
  170 stop ' BWACO: ERROR - Unable to open bwaco.cfg!'
c
c      format statements
 1010 format(i1)
 1020 format(i2)
 1030 format(i1,2i2)
 2010 format(//'  BWACO CONFIGURATION MENU')
 2020 format( /'  Default Display:   ',a24)
 2030 format(  '  Default Printer:   ',a24)
 2040 format( /'  Enter 1 to change the default display')
 2050 format(  '        2 to change the default printer')
 2060 format(  '        3 to save the configuration')
 2071 format(  '        0 to proceed to the Main Menu')
 2072 format(  '        0 to return to the Main Menu')
 2073 format(  '        0 to return to the Plotting Menu')
 2080 format(  '        '\)
 2110 format
     +(/'  1: CGA Mode  4 320x200/4      2: CGA Mode 5  320x200/4')
 2120 format
     +( '  3: CGA Mode  6 640x200/BW     4: Hercules    720x348/BW')
 2130 format
     +( '  5: EGA Mode 13 320x200/16     6: EGA Mode 14 640x200/16')
 2140 format
     +( '  7: EGA Mode 15 640x350/BW     8: EGA Mode 16 640x350/16')
 2150 format
     +( '  9: VGA Mode 17 640x480/BW    10: VGA Mode 18 640x480/16')
 2160 format
     +( ' 11: ATI Mode 81 640x480/16    12: ATI Mode 82 752x410/16')
 2170 format(' 13: ATI Mode 83 800x560/16')
 2180 format(/' Enter the new display index: '\)
 2210 format
     +(/' 14: Single-density printer    15: Double-density printer')
 2220 format
     +( ' 16: Quad-density printer      17: Epson Print  plot mode')
c2230 format
c    +( ' 18: HP Plotter (Scaling)      19: HP Plotter  (Absolute)')
 2240 format
     +( ' 20: HP Laser Jet  (75 dpi)    21: HP Laser Jet  (100 dpi)')
 2250 format
     +( ' 22: HP Laser Jet  (150 dpi)   23: HP Laser Jet  (300 dpi)')
 2260 format
     +( ' 24: HP Think Jet  Lo Dens     25: HP Think Jet  Hi Dens')
 2270 format
     +( ' 26: HP Paint Jet  Lo Dens     27: HP Paint Jet  Hi Dens')
 2280 format
     +( ' 28: Postscript Printer        29: Encapsulated Postscript')
 2290 format(/' Enter the new printer index: '\)
 3010 format(/' rcfg called')
 3020 format('    display = ',i2)
 3030 format('    printer = ',i2)
 3040 format('    configuration saved')
c
      end
c---------------------------------------------------------------------
```

Figure A-24. Subroutine *rcfg* from the interface module.

```
c______________________________________________________________________
c
      subroutine savn
c
c      saves new data to file
c
      common/icfg/ icomb,iscrn,iprnt
      common/totl/ wtotl
      common/ulab/ ulabl
      common/stkn/ nstak
      common/stkp/ xstak(100),ystak(100),zstak
      common/stkd/ idonr(100),nidnr
      common/regn/ xcent,ycent,rinhb,xmini,xmaxi,ymini,ymaxi
      common/sttn/ nstat
      common/sttb/ lblok,nblok
      common/sttp/ xstat(120),ystat(120),zstat
      common/sttv/ icoal(3600),pstat(3600),wstat(3600)
      common/mxmn/ xmin,xmax,ymin,ymax
c
      character*80 ulabl
c
c      log call
      write(8,3010)
c
      open(7,file='bwaco.dat',form='unformatted',err=110)
      write(7) ulabl
      write(7) nstat,lblok,nblok,nstak,nidnr
      write(7) wtotl,xmin,xmax,ymin,ymax
      write(7) xcent,ycent,rinhb
      write(7) (xstak(i),i=1,nstak)
      write(7) (ystak(i),i=1,nstak)
      write(7) (idonr(i),i=1,nstak)
      write(7) (xstat(i),i=1,lblok)
      write(7) (ystat(i),i=1,nblok)
      write(7) (pstat(i),i=1,nstat)
      write(7) (wstat(i),i=1,nstat)
      write(7) (icoal(i),i=1,nstat)
      close(7)
c
      return
c
c      error detected
  110 stop ' BWACO: ERROR - Unable to open bwaco.dat!'
c
c      format statement
 3010 format(/' savn called')
c
      end
c
c______________________________________________________________________
c______________________________________________________________________
c
      subroutine geto
c
c      gets old plotting data from file
c
      common/icfg/ icomb,iscrn,iprnt
      common/totl/ wtotl
      common/ulab/ ulabl
      common/stkn/ nstak
      common/stkp/ xstak(100),ystak(100),zstak
      common/stkd/ idonr(100),nidnr
      common/regn/ xcent,ycent,rinhb,xmini,xmaxi,ymini,ymaxi
      common/sttn/ nstat
      common/sttb/ lblok,nblok
      common/sttp/ xstat(120),ystat(120),zstat
      common/sttv/ icoal(3600),pstat(3600),wstat(3600)
      common/mxmn/ xmin,xmax,ymin,ymax
c
      character*80 ulabl
c
c      log call
      write(8,3010)
c
      open(7,file='bwaco.dat',form='unformatted',status='old',err=110)
      read(7,end=130,err=120) ulabl
      read(7,end=130,err=120) nstat,lblok,nblok,nstak,nidnr
      read(7,end=130,err=120) wtotl,xmin,xmax,ymin,ymax
      read(7,end=130,err=120) xcent,ycent,rinhb
      read(7,end=130,err=120) (xstak(i),i=1,nstak)
      read(7,end=130,err=120) (ystak(i),i=1,nstak)
      read(7,end=130,err=120) (idonr(i),i=1,nstak)
      read(7,end=130,err=120) (xstat(i),i=1,lblok)
      read(7,end=130,err=120) (ystat(i),i=1,nblok)
      read(7,end=130,err=120) (pstat(i),i=1,nstat)
      read(7,end=130,err=120) (wstat(i),i=1,nstat)
      read(7,end=130,err=120) (icoal(i),i=1,nstat)
      close(7)
c
      return
c
c      errors detected
  110 stop ' BWACO: ERROR - Unable to open bwaco.dat!'
  120 stop ' BWACO: ERROR - Unable to read bwaco.dat!'
  130 stop ' BWACO: ERROR - Premature end of bwaco.dat!'
c
c      format statement
 3010 format(/' geto called')
c
      end
c______________________________________________________________________
```

Figure A-25. Subroutines *savn* and *geto* from the interface module.

Subroutine *savn*, shown in Figure A-25, saves the results of a computation to a file.

Subroutine *geto*, shown in Figure A-25, gets the results of a previously run computation from a file.

Subroutine *pmen*, shown in Figure A-26, processes requests from the plotting menu when a computation is complete or when the user requests plotting from the main menu.

Subroutine *cnfg*, shown in Figure A-27, obtains the default configuration parameters from the file *bwaco.cfg* or calls *rcfg* if *bwaco.cfg* does not exist.

Subroutine *cnst*, shown in Figure A-28, sets the required constants.

Subroutine *mmen*, shown in Figure A-26, processes requests from the main menu.

Program *bwac*, shown in Figure A-27, is the main program. It configures the session with calls to *cnfg* and *cnst*, and initiates the main menu.

```
c
c-----------------------------------------------------------------------
      subroutine pmen
c
c      solicits and processes a user request from the plotting menu
c
c      local variables:
c       irqst = user request index
c       iptyp = index of requested plot type
c       idevi = index of requested device
c
c      log call
      write(8,3010)
c
c      restore original units
      call rest
c
c      solicit user request
  110 call spmn(irqst,iptyp,idevi)
c
      if(irqst.gt.0.and.irqst.lt.5) then
c
c         process plotting request
        call plts(iptyp,idevi)
c
        go to 110
c
      endif
c
      if(irqst.eq.5) then
c
c         process reconfiguration request
        call rcfg(0,3)
        call savn
c
        go to 110
c
      endif
c
      return
c
c      format statement
 3010 format(/' pmen called')
c
      end
c
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      subroutine mmen
c
c      solicits and processes a user request from the main menu
c
c      local variable:
c       irqst = user request index
c
c      log call
      write(8,3010)
c
c      solicit user request
      iscur=0
  110 call smmn(iscur,irqst)
c
      if(irqst.eq.1) then
c
c         process a request to run a problem
        call exec
        call savn
        call pmen
        iscur=1
c
      endif
c
      if(irqst.eq.2) then
c
c         process a request to plot a problem
        call geto
        call pmen
        iscur=1
c
      endif
c
      if(irqst.eq.3) then
c
c         process a request for session reconfiguration
        call rcfg(0,2)
c
      endif
c
      if(irqst.ne.0) go to 110
c
      return
c
c      format statement
 3010 format(/' mmen called')
c
      end
c-----------------------------------------------------------------------
```

Figure A-26. Subroutines *pmen* and *mmen* from the interface module.

```
c-----------------------------------------------------------------------
c
      program bwac
c
c      is the BWACO main program
c
c      report the BWACO version
      write(6,2010)
c
c      open the session log
      open(8,file='bwaco.log',form='formatted',err=110)
      write(8,3010)
c
c      configure the session
      call cnfg
c
c      define the constants
      call cnst
c
c      run the main menu
      call mmen
c
c      close the session log
      write(8,3020)
      close(8)
c
c      terminate execution
      stop ' BWACO: Session Terminated.'
c
c      error detected
  110 stop ' BWACO: ERROR - Unable to open bwaco.log!'
c
c      format statement
 2010 format(//' BWACO: May 1994')
 3010 format(' BWACO Session Log')
 3020 format(/' termination requested')
c
      end
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
c
      subroutine cnfg
c
c      configures the session
c
c      configuration variables;
c       icomb = pressure combination algorithm switch
c       iscrn = default display switch
c       iprnt = default printer switch
c
      common/icfg/ icomb,iscrn,iprnt
c
c      log call
      write(8,3010)
c
c      read data from the configuration file
      open(7,file="bwaco.cfg",form="formatted",status="old",err=110)
      read(7,1010,end=110,err=110) icomb,iscrn,iprnt
      close(7)
      go to 120
c
c      generate default configuration file
  110 icomb=2
      iscrn=10
      iprnt=28
      call rcfg(1,1)
c
  120 return
c
c      format statements
 1010 format(i1,2i2)
 3010 format(/' cnfg called')
c
      end
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
c
      subroutine exec
c
c      sets up and executes the coalescence computation
c
c      log call
      write(8,3010)
c
c      read problem input
      call inpt
c
c      define the stacks
      call stak
c
c      define the stations
      call stns
c
c      make the coalescence computation
      call copt
c
      return
c
c      format statement
 3010 format(/' exec called')
c
      end
c-----------------------------------------------------------------------
```

Figure A-27. Program *bwac* and subroutines *cnfg* and *exec* from the interface module.

```
c_______________________________________________________________________
c
      subroutine cnst
c
c      defines the constants
c
      common/conf/ thrd,tpi
      common/conv/ okt,ocm,odsc,ocmkt3
      common/coni/ gamma,gmone,gpone
      common/cona/ p1,c1,r1,t1
      common/conp/ a0p,b0p2,c0p,a1p,b1p,r0p
      common/cons/ r0s,c0s,d0s
      common/conk/ ak0,ak(8)
c
c      log call
      write(8,3010)
c
c      fixed constants
      thrd=1./3.
      tpi=8.*atan(1.)
      twthrd=2.**thrd
c
c      conversion factor constants
      okt=2.e6
      ocm=0.01
      odsc=68947.6
      ocmkt3=2.604004e-4*twthrd
c
c      ideal gas constants
      gamma=1.404574
      gmone=gamma-1.
      gpone=gamma+1.
c
c      sea-level atmospheric constants
      call at62(0., p1,c1,r1,t1)
c
c      positive phase duration constants
c       (time in s, distance in cm, weight in kt)
      a0p =4.313976E-05
      b0p2=6.261821E+08
      c0p =3.350000E-01
      a1p =5.361303E+12
      b1p =3.507575E+00
      r0p =4.490000E+03
c
c      shock arrival time constants
c       (time in s, distance in cm, weight in kt)
      r0s=1.6195e4
      c0s=3.3897e4
      d0s=1.5000e4
c
c      kingery overpressure fit constants
c      (pressure in lb/in**2, time in s, distance in ft, weight in lb)
      ak0  = 7.0452041
      ak(1)=-1.6277561
      ak(2)=-0.27399088
      ak(3)=-6.5973136e-2
      ak(4)= 6.5412563e-3
      ak(5)= 4.8236359e-2
      ak(6)=-2.0072553e-2
      ak(7)= 3.0190449e-3
      ak(8)=-1.5984026e-4
c
      return
c
c      format statement
 3010 format(/' cnst called')
c
      end
c_______________________________________________________________________
```

Figure A-28. Subroutine *cnst* from the interface module.

INTENTIONALLY LEFT BLANK.

APPENDIX B:

BWACO USER'S MANUAL

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

INTRODUCTION

This manual provides background information on the BWACO methodology, outlines hardware and software requirements and directs installation of the code. The BWACO Tutorial gives instructions for running sample problems and creating and running your own problems. In addition, the BWACO Reference describes all BWACO functions and features in alphabetical order. Finally, a detailed description of BWACO, listing most of the source code, is provided. Users who experience difficulty should contact the authors at (410) 278-6214/6241.

BACKGROUND

The BWACO program was written to predict regions of blast wave coalescence associated with the sequential mass detonation of distributed ammunition stacks positioned as specified by the user.

It accomplishes this by applying the assumption that communication of mass detonation between stacks is controlled by a user-supplied propagation velocity (which might represent the velocity of fragments from the donor stack). BWACO uses this propagation velocity to determine the order and timing of detonation of the stacks. It recognizes that the identity of the initial donor is not generally known and, unless a single initial donor has been specified by the user, considers all possible initial donors in order to predict the worst-case environment at any point.

Then, under the assumption that blast waves propagate independently of one another, BWACO applies the criterion that coalescence occurs wherever and whenever the shock associated with one blast wave encroaches into the positive overpressure phase of another blast wave. This assumption generally causes premature identification of coalescence. When coalescence is detected at a point, it determines the combined peak overpressure by first combining, at their center of charge, the stacks producing the waves which coalesce and then computing the peak overpressure associated with the combined stacks at that point. This method is suitable, at least, for closely spaced stacks.

BWACO graphically reports regions of coalescence, showing the effective total explosive weight and the peak overpressure throughout the region of interest.

In applying BWACO, it is important to remain aware of limitations imposed by the assumptions. The accuracy of BWACO has not been assessed except in the case of closely spaced charges. As the separation between stacks is increased, the accuracy, especially that of the peak overpressure predictions, is expected to decrease.

Detailed background information can be found in the references.

HARDWARE AND SOFTWARE REQUIREMENTS

BWACO should run in the DOS environment on any personal computer with an 80?86 microprocessor. However, to minimize the running times for large problems, a high-speed computer with at least an 80486 microprocessor is recommended.

An editor (such as the DOS editor) that produces pure ASCII text (without hidden characters) is required to create problem input files. A word processing program should not be used unless the file can be saved in ASCII format.

BWACO's graphics supports a number of different video displays and printers. The displays are listed in the following table:

Type	Mode	Resolution	Colors
CGA	4	320x200	4
CGA	5	320x200	4
CGA	6	640x200	BW
Hercules		720x348	BW
EGA	13	320x200	16
EGA	14	640x200	16
EGA	15	640x350	BW
EGA	16	640x350	16
VGA	17	640x480	BW
VGA	18	640x480	16
ATI	81	640x480	16
ATI	82	752x410	16
ATI	83	800x560	16

The following printers are supported:

```
single-density printer
double-density printer
quad-density printer
Epson printer (plot mode)
Postscript printer
Hewlett-Packard Laser Jet
Hewlett-Packard Think Jet
```

Hard copies requested during a BWACO session are automatically sent to the printer at the end of the session.

INSTALLATION

Please follow these installation instructions very carefully. Before installing BWACO, create a directory to hold the BWACO files. For example, you can create the directory **bwaco** by entering the following line:

```
>mkdir bwaco
```

Make that directory the current directory.

```
>cd bwaco
```

Insert the BWACO installation diskette into the appropriate drive. If you inserted it into drive `a:`, enter the following line:

```
>a:install a
```

If you inserted it into drive `b:`, enter the following line:

```
>b:install b
```

This will cause four files to be copied to your new directory, after which the installation is complete.

BWACO TUTORIAL

Getting Started

BWACO always looks for input in a file with the extension **inp**. A sample input file has been supplied and running this problem before trying your own is recommended. The file **three.inp** describes three 25,000-lb stacks of ammunition, separated by 200 meters, lying at the vertices of an equilateral triangle. To start a BWACO session and run this problem, simply enter the following line:

```
>bc three
```

The first time you run BWACO the Configuration Menu will appear. This menu, which is similar to the other BWACO menus, requires you to enter a single-digit number to select one of the menu options:

```
BWACO CONFIGURATION MENU

Default Display:  VGA Mode 18 640x480/16
Default Printer:  Postscript Printer

Enter 1 to change the default display
      2 to change the default printer
      0 to proceed to the Main Menu
```

The Configuration Menu allows you to tell BWACO which display and printer are installed on your system. The default display is **VGA Mode 18** and the default printer is **Postscript Printer**. If your display or printer is different, make changes now. If you do not specify a display or printer, BWACO will save and use these defaults. To change the display or printer, simply enter either 1 or 2 and make your selection from the list displayed. BWACO will remember your specifications in all subsequent sessions. To leave the menu, simply strike the enter key. Typing 0 is not necessary. After you have completed the initial configuration, the Main Menu will appear:

```
BWACO MAIN MENU

Enter 1 to run the current problem
      2 to reconfigure the session
      0 to end the session
```

To run the problem configuration described in three.inp, enter 1. You will have to wait some period of time while the problem runs. BWACO will keep you informed of what it is doing. When the computation is complete, BWACO will tell you how long it took to run on your system and the Plotting Menu will appear:

```
BWACO PLOTTING MENU

Enter 1 to plot the Coalescence Map (Screen)
      2 to plot the Coalescence Map (Printer)
      3 to plot the Peak Overpressure Map (Screen)
      4 to plot the Peak Overpressure Map (Printer)
      5 to reconfigure the session
      0 to return to the Main Menu
```

You can read about the Coalescence and Peak Overpressure Maps in the reference section of this manual. Produce the Coalescence Map on your screen by entering 1. Strike any key to return to the Plotting Menu. Enter 2 to obtain a hard copy. All hard copies requested during a session will be printed when the session is terminated. Try the Peak Overpressure Map in a similar fashion. If you experience problems with plotting, you may have made the wrong configuration selections. You can correct them by entering 5 from the Plotting Menu.

When you are finished plotting, return to the Main Menu:

```
BWACO MAIN MENU

Enter 1 to run the current problem
      2 to plot the current problem
      3 to reconfigure the session
      0 to end the session
```

This menu now provides the additional option of plotting the results of the problem without rerunning. End the session by striking the **enter** key. You may receive a query regarding your printer. If so, simply strike **enter** again. Any hard copies you requested will print at this time.

Creating Your Own Input Files

BWACO requires its input file to have a very specific format. Before creating a new input file, let's look at the format of the file **three.inp**.

```
Three 25-klb Stacks, 200-m Separation (three.inp)
|--Xmin--||--Xmax--||--Ymin--||--Ymax--|
     -700.       700.      -700.       700.
|---Xs---||---Ys---||---Ws---||---Vs---|    D
      100.    -57.74    25000.    2500.0    1
     -100.    -57.74    25000.    2500.0    1
        0.    115.48    25000.    2500.0    1
```

The first line of this file contains a descriptive problem title of the user's choice. It may contain up to 80 characters. It is used to label the plots. As a convenience, we have included the input file name in parentheses at the end of the title. The second line is a ruler line delimiting the fields in which the bounds of the region of interest are placed. These bounds, which appear on the third line, determine the region in which the stations are distributed. The fourth line is another ruler line specifying the fields in which stack data is placed. The three stacks are described, one to a line, on the remaining lines. The stack coordinates, the TNT equivalent explosive weight, and the propagation velocity are specified. In addition, the D (initial donor) switch is set to 1 to ensure that each stack may act as an initial donor.

Let's see how problem input is generated. The net explosive weight for each stack should be expressed in equivalent pounds of TNT, distances in meters and propagation velocities in meters per second. Suppose we have four 25,000-lb ammunition stacks arranged as shown in Figure B-1. First, we must define a Cartesian coordinate system. The orientation and origin location of our coordinate system is arbitrary and may be chosen to suit our convenience. One possibility, along with the coordinates of each stack, is shown in Figure B-2. In addition to this, we also have to know the propagation velocity associated with each stack. Since no better guidance is presently available, we have fixed this at 2,500 m/s.

We now know enough to generate the input file. In order to create your own problem input files, you will need to use an editor that produces pure ASCII files (without hidden characters), such as the DOS editor. A word processing program should not be used. As a convenience, an input template has been included with BWACO. To start out, simply copy this file to the new input file, as follows:

```
>copy template.inp four.inp
```

Then, edit the file **four.inp**. Initially, it looks like this:

```
|--Xmin--||--Xmax--||--Ymin--||--Ymax--|

|---Xs---||---Ys---||---Ws---||---Vs---|     D
```

First we'll add a descriptive title:

```
Four 25-klb Stacks, 200-m Separation (four.inp)
|--Xmin--||--Xmax--||--Ymin--||--Ymax--|

|---Xs---||---Ys---||---Ws---||---Vs---|     D
```

Next, since we don't know how big the region of interest should be, we'll let BWACO determine its limits by specifying X and Y bounds which are equal:

```
Four 25-klb Stacks, 200-m Separation (four.inp)
|--Xmin--||--Xmax--||--Ymin--||--Ymax--|
       0.        0.        0.        0.
|---Xs---||---Ys---||---Ws---||---Vs---|     D
```

Finally, we'll specify the stack coordinates, weights and propagation velocities, setting the initial donor switch to 1 for each stack:

```
Four 25-klb Stacks, 200-m Separation (four.inp)
|--Xmin--||--Xmax--||--Ymin--||--Ymax--|
       0.        0.        0.        0.
|---Xs---||---Ys---||---Ws---||---Vs---|     D
    -100.     -100.    25000.     2500.      1
     100.     -100.    25000.     2500.      1
    -100.      100.    25000.     2500.      1
     100.      100.    25000.     2500.      1
```

This completes the generation of the input file **four.inp**. Try running this problem:

```
>bc four
```

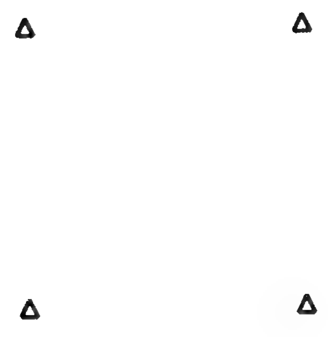


Figure B-1. Arrangement of four 25,000-lb ammunition stacks.

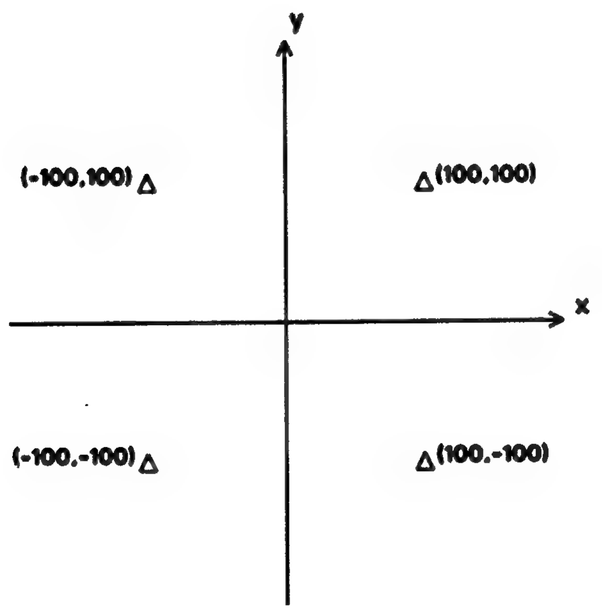


Figure B-2. A coordinate system for this arrangement.

The plots produced will be somewhat distorted. This occurs because the algorithm used to extend the axes to a natural termination point treats the X and Y axes differently. To eliminate the distortion, it is necessary to rerun the problem after modifying the input file to specify region limits:

```
Four 25-klb Stacks, 200-m Separation (four.inp)
|--Xmin--||--Xmax--||--Ymin--||--Ymax--|
     -800.      800.     -800.      800.
|---Xs---||---Ys---||---Ws---||---Vs---|    D
     -100.     -100.    25000.     2500.    1
      100.     -100.    25000.     2500.    1
     -100.      100.    25000.     2500.    1
      100.      100.    25000.     2500.    1
```

With this modification (which is not always necessary when allowing BWACO to determine the extent of the region of interest) undistorted plots, as shown in Figures B-3 and B-4, are produced.

Obtaining Plots From Earlier Computations

We can easily obtain new plots from the problem described in **three.inp** run earlier. Be sure to wait until all previously requested plots have printed, otherwise the files may be deleted before they print. Simply enter the following line:

```
>bc three
```

The Main Menu appears as follows:

```
BWACO MAIN MENU

Enter 1 to run the current problem
      2 to plot the previous problem
      3 to reconfigure the session
      0 to end the session
```

The previously obtained results may be plotted by entering **2** and making the desired selections from the Plotting Menu. End the session by striking **enter** twice.

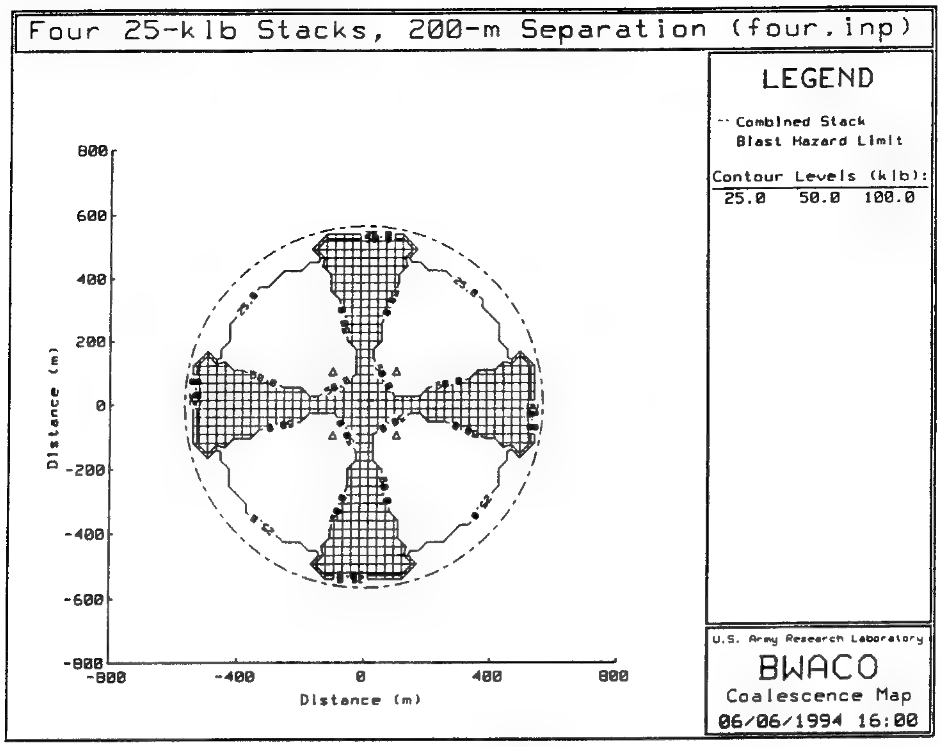


Figure B-3. Coalescence Map.

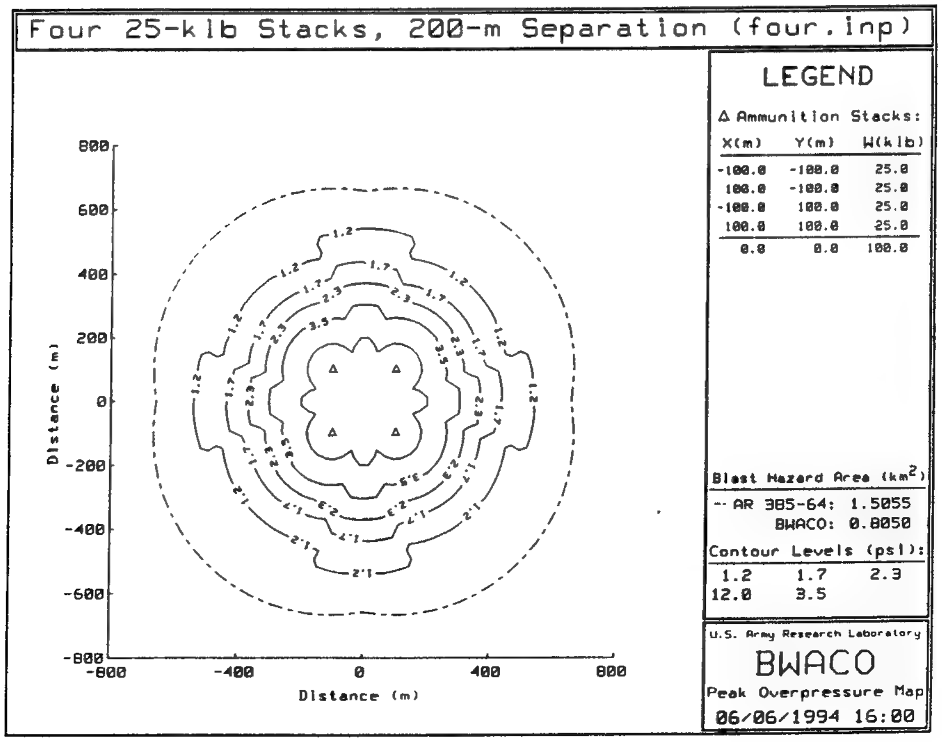


Figure B-4. Peak Overpressure Map.

Coalescence Map

An example of a Coalescence Map is shown in Figure B-3. Regions in which coalescence is detected are hatched and explosive weight contours are drawn. Each contour is labeled to indicate the total explosive weight contributing to the worst-case blast environment within that contour. The outermost contour follows the 0.9 or 1.2 psi overpressure limit for inhabited buildings defined in Army Regulation (AR) 385-64. For comparison purposes, a circle, centered at the center of charge of all the stacks and having the inhabited building distance for the combined explosive weight of all the stacks as its radius, is also plotted. As many contour levels as space permits are listed in the legend. (See **Peak Overpressure Map** and **Plotting Distortion**.)

Contour Levels

The contour levels used in the Coalescence Map are listed in the legend (space permitting). These levels describe the total explosive weight contributing to the worst-case blast environment within the contour. The contour levels used in the Peak Overpressure Map are those specified in AR 385-64 for the protection of a variety of assets.

Files and Program Execution

The file **`bc.bat`** is the DOS batch file that executes BWACO and manages its files. The file **`bwaco.exe`** is the executable BWACO program. The files **`*.inp`** are the user's problem specification input files (where * represents the user's descriptive file name). (See **Input Files**.) The files **`*.dat`** contain the plotting data, enabling BWACO to plot results without rerunning problems. The files **`*.log`** contain logs of BWACO's most recent session pertaining to the problem identified by *. The user starts a BWACO session by entering **`bc *`**. The batch file first deletes any old plotting files. For this reason, a BWACO session should never be started until all plots from the previous session have printed. Next, it copies **`*.inp`** and **`*.dat`** (if it exists) to **`bwaco.inp`** and **`bwaco.dat`**, respectively. It then executes **`bwaco.exe`**, which creates the files **`bwaco.dat`** and **`bwaco.log`**. On termination of the session, the batch file deletes **`bwaco.inp, moves bwaco.dat`** and **`bwaco.log`** to **`*.dat`** and **`*.log`**, respectively and sends all plotting files to the printer using the **`print`** command.

Inhabited Building Limit

AR 385-64 defines permissible overpressure exposure levels for a variety of assets. The lowest such levels are those commonly associated with inhabited buildings (0.9 to 1.2 psi). The levels are also given as scaled distances using classical blast scaling laws. The minimum distance is further limited by consideration of hazardous fragment density. BWACO combines all the stacks by summing their explosive weights and determining their center of charge. Using the combined weight, it computes the inhabited building distance. Using the center of charge, it plots the combined inhabited building circle on the Coalescence Map. On the Peak Overpressure Map, it plots a contour produced by centering inhabited building distance circles at each stack. This represents the method currently required by the regulation and allows comparison with BWACO's predictions.

Initial Donors

When detonation propagates from one stack to another, the first stack of such a pair is referred to as the donor. The first of all the stacks to detonate is called the initial donor. BWACO allows the user to specify which stacks may act as initial donors. It is not generally possible to predict which ammunition stack will act as an initial donor when a mass-detonation accident occurs. Therefore, in most applications, the user should specify that all stacks may act as initial donors. BWACO will issue a warning whenever any stacks may not act as initial donors. (See **Input Files**.)

Input Files

An editor that produces pure ASCII files (without hidden characters), such as the DOS editor, must be used to create problem input files. A word processing program should not be used unless the file can be saved in ASCII format. BWACO looks for input in a file with the extension `inp`. A descriptive file name (up to eight characters) may be used. The file must conform to the format (see **Template**) shown in the following example:

```
Four 25-klb Stacks, Separation=200 m (four.inp)
|--Xmin--||--Xmax--||--Ymin--||--Ymax--|
       0.        0.        0.        0.
|---Xs---||---Ys---||---Ws---||---Vs---|    D
    -100.     -100.    25000.     2500.    1
     100.     -100.    25000.     2500.    1
    -100.      100.    25000.     2500.    1
     100.      100.    25000.     2500.    1
```

The first line of the file contains a descriptive problem title of the user's choice. It may contain up to 80 characters. It is used to label the plots, centered in the plot label box at the top of both the Coalescence and Peak Overpressure Maps. The second line is a ruler line delimiting the fields in which the bounds of the region of interest are placed. These bounds, which appear on the third line, determine the region in which the stations are distributed. When the input bounds for either coordinate axis are equal (Xmin=Xmax or Ymin=Ymax), BWACO computes appropriate bounds. The fourth line is another ruler line specifying the fields in which stack data is placed. Up to 100 stacks are described, one to a line, on the remaining lines. The stack coordinates, net equivalent explosive weight, and propagation velocity are specified. In addition, the D (initial donor) switch is set to 0 or 1 to specify whether or not each stack may act as an initial donor. For most problems, the initial donor switch should be set to 1 for all stacks. (See **Initial Donors**.)

Menus

Menu selections are made by entering a single-digit number at the prompt. A menu is exited by entering 0 or its equivalent, simply striking **enter**.

The Configuration Menu allows selection of the appropriate video display and printer. It may appear in any of the following three forms:

```
BWACO CONFIGURATION MENU

Default Display:  VGA Mode 18 640x480/16
Default Printer:  Postscript Printer

Enter 1 to change the default display
      2 to change the default printer
      0 to proceed to the Main Menu
```

```
BWACO CONFIGURATION MENU

Default Display:  VGA Mode 18 640x480/16
Default Printer:  Postscript Printer

Enter 1 to change the default display
      2 to change the default printer
      0 to return to the Main Menu
```

```
BWACO CONFIGURATION MENU

Default Display:  VGA Mode 18 640x480/16
Default Printer:  Postscript Printer

Enter 1 to change the default display
      2 to change the default printer
      0 to return to the Plotting Menu
```

The Main Menu allows selection of the problem execution, plotting or reconfiguration functions, as available. It may appear in any of the following three forms:

```
BWACO MAIN MENU

Enter 1 to run the current problem
      2 to reconfigure the session
      0 to end the session
```

```
BWACO MAIN MENU

Enter 1 to run the current problem
      2 to plot the current problem
      3 to reconfigure the session
      0 to end the session
```

```
BWACO MAIN MENU

Enter 1 to run the current problem
      2 to plot the previous problem
      3 to reconfigure the session
      0 to end the session
```

The Plotting Menu allows selection of Coalescence or Peak Overpressure Maps on either the screen or the printer. It appears as follows:

```
BWACO PLOTTING MENU

Enter 1 to plot the Coalescence Map (Screen)
      2 to plot the Coalescence Map (Printer)
      3 to plot the Peak Overpressure Map (Screen)
      4 to plot the Peak Overpressure Map (Printer)
      5 to reconfigure the session
      0 to return to the Main Menu
```

Peak Overpressure Map

An example of a Peak Overpressure Map is shown in Figure B-4. Peak overpressure contours are labeled to indicate the worst-case values at each point within the region of interest. The contour levels, which are listed in the legend, are those specified in AR 385-64 for the protection of a variety of assets. For comparison purposes, an additional contour representing the application of current regulatory requirements is also shown. The positions and equivalent explosive weights of the stacks as well as the center of charge and total equivalent explosive weight are also shown in the legend. (See **Coalescence Map** and **Plotting Distortion**.)

Plotting Distortion

Plotting distortion occurs when the algorithm used to extend the axes to a natural termination point treats the X and Y axes differently. To eliminate the distortion, it is necessary to rerun the problem after modifying the input file to specify different region limits. (See **Coalescence Map** and **Peak Overpressure Map**.)

Printers

The following printers are supported:

```
single-density printer
double-density printer
quad-density printer
Epson printer (plot mode)
Postscript printer
Hewlett-Packard Laser Jet
Hewlett-Packard Think Jet
```

Region of Interest

The region of interest is the region covered by the Coalescence and Peak Overpressure Maps. It is specified by giving values of Xmin, Xmax, Ymin, and Ymax in the user's input file. If the user specifies values of Xmin and Xmax which are equal, BWACO will compute the range of the region of interest in the X direction. An analogous algorithm applies in the Y direction. (See **Stations**.)

Stations

When running a problem, BWACO positions 3,600 stations within the region of interest. These are not generally apparent to the user. Coalescence is checked at each of these stations for each initial donor. (See **Region of Interest**.)

Template

The file **`template.inp`** is a template for preparing input files. It can be copied as the first step in preparing a new input file. It includes the two ruler lines and appears as follows:

```
|--Xmin--||--Xmax--||--Ymin--||--Ymax--|
|---Xs---||---Ys---||---Ws---||---Vs---|     D
```

(See **Input Files**.)

Video Displays

The supported displays are listed in the following table:

Type	Mode	Resolution	Colors
CGA	4	320x200	4
CGA	5	320x200	4
CGA	6	640x200	BW
Hercules		720x348	BW
EGA	13	320x200	16
EGA	14	640x200	16
EGA	15	640x350	BW
EGA	16	640x350	16
VGA	17	640x480	BW
VGA	18	640x480	16
ATI	81	640x480	16
ATI	82	752x410	16
ATI	83	800x560	16

NO. OF COPIES	ORGANIZATION
2	ADMINISTRATOR DTIC ATTN DTIC DDA CAMERON STATION ALEXANDRIA VA 22304-6145
1	CDR USAMC ATTN AMCAM 5001 EISENHOWER AVE ALEXANDRIA VA 22333-0001
1	DIR USARL ATTN AMSRL OP SD TA RECORDS MGMT 2800 POWDER MILL RD ADELPHI MD 20783-1145
3	DIR USARL ATTN AMSRL OP SD TL TECH LIB 2800 POWDER MILL RD ADELPHI MD 20783-1145
1	DIR USARL ATTN AMSRL OP SD TP TECH PUBS BR 2800 POWDER MILL RD ADELPHI MD 20783-1145
2	CDR US ARMY ARDEC ATTN SMCAR TDC PICATINNY ARSNL NJ 07806-5000
1	DIR BENET LABS ATTN SMCAR CCB TL WATERVLIET NY 12189-4050
1	DIR USA ADVANCED SYSTEMS R&A OFC ATTN AMSAT R NR MS 219 1 AMES RESEARCH CENTER MOFFETT FIELD CA 94035-1000
1	CDR US ARMY MICOM ATTN AMSMI RD CS R DOC REDSTONE ARSNL AL 35898-5010
1	CDR US ARMY TACOM ATTN AMSTA JSK ARMOR ENG BR WARREN MI 48397-5000

NO. OF COPIES	ORGANIZATION
1	DIR USA TRADOC ANALYSIS CMD ATTN ATRC WSR WSMR NM 88002-5502
1	CMDT US ARMY INFANTRY SCHOOL ATTN ATSH CD SECURITY MGR FORT BENNING GA 31905-5660
	ABERDEEN PROVING GROUND
2	DIR USAMSAA ATTN AMXSY D AMXSY MP H COHEN
1	CDR USATECOM ATTN AMSTE TC
1	DIR USAERDEC ATTN SCBRD RT
1	CDR USACBDCOM ATTN AMSCB CII
1	DIR USARL ATTN AMSRL SL I
5	DIR USARL ATTN AMSRL OP AP L

NO. OF COPIES	ORGANIZATION
1	DIRECTOR US ARMY TECHNICAL CENTER FOR EXPLOSIVE SAFETY ATTN: SMCAC-ESL/C DOYLE SAVANNA IL 61074-9639
3	US ARMY WATERWAYS EXPERIMENT STATION ATTN: CEWES-SE/ K DAVIS M FORD C JOACHIM 3909 HALLS FERRY RD VICKSBURG MS 39180-6199
3	CHAIRMAN DEPARTMENT OF DEFENSE EXPLOSIVE SAFETY BOARD ATTN: DDESB-KT/ J WARD (2 CP) C CANADA 2461 EISENHOWER AVE ALEXANDRIA VA 22331-0600
1	DEFENSE AMMUNITION LOGISTICS ACTIVITY ATTN: AMCPM-AL/R ROSSI PICATINNY ARSENAL NJ 07806-5000
1	OFFICER IN CHARGE WHITE OAK LABORATORY NAVAL SURFACE WARFARE CENTER ATTN: R15/M SWISDAK 10901 NEW HAMSPHIRE AVE SILVER SPRING MD 20902-5000
2	NAVAL FACILITIES ENGINEERING SERVICES CENTER ATTN: CODE LS1/ J TANCRETO R MURTHE PORT HUENEME CA 93043
2	MAXWELL S-CUBED DIVISION ATTN: C NEEDHAM S HIKIDA 2501 YALE BLVD SE SUITE 300 ALBUQUERQUE NM 87106-4215

NO. OF COPIES	ORGANIZATION
3	SOUTHWEST RESEARCH INSTITUTE ATTN: P BOWLES C OSWALD E ESPARZA 6220 CULEBRA ROAD PO DRAWER 28510 SAN ANTONIO TX 78228-0510
1	DIRECTOR EXPLOSIVES HAZARD REDUCTION DIRECTORATE ATTN: J JENUS ASC/YOCO (EHR) EGLIN AFB FL 32542
2	US ARMY CORPS OF ENGINEERS ATTN: CEHND-ED-CS/ R WRIGHT R HASSE PO BOX 1600 HUNTSVILLE AL 35807-4301
3	COMMANDER DEFENSE NUCLEAR AGENCY ATTN: T KENNEDY D LINGER G ULLRICH WASHINGTON DC 20305-1000
1	COMMANDER FIELD COMMAND DNA ATTN: FCTTS (MCCRORY) KIRTLAND AFB NM 87117-5669
1	HQDA (SARD) ATTN: MR MLARCHIK WASHINGTON DC 20310-0001
1	COMMANDER US ARMY NUCLEAR AND CHEMICAL AGENCY ATTN: MONA-NU 7150 HELLER LOOP SUITE 101 SPRINGFIELD VA 22150-3198
1	ABERDEEN RESEARCH CENTER ATTN: N ETHRIDGE PO BOX 548 ABERDEEN MD 21001

NO. OF COPIES	ORGANIZATION
1	CARPENTER RESEARCH CORPORATION ATTN: J CARPENTER PO BOX 2490 ROLLING HILLS ESTATES CA 90274
1	LOGICON R&D ASSOCIATES ATTN: G GANONG PO BOX 9335 ALBUQUERQUE NM 87119
1	TECH REPS INC ATTN: F MCMULLAN 5000 MARBLE NE SUITE 222 ALBUQUERQUE NM 87110
	ABERDEEN PROVING GROUND
20	DIR, USARL ATTN: AMSRL-WT-T/W MORRISON AMSRL-WT-TB/ R FREY O LYMAN T WATSON F GREGORY V BOYLE W HILLSTROM E MCDOUGAL W LAWRENCE K BENJAMIN T DORSEY AMSRL-WT-TD/ A DAS GUPTA J SANTIAGO AMSRL-WT-NC/ R LOTERRO R RALEY J SULLIVAN S SCHRAML AMSRL-CI-CA/ A CELMINS C ZOLTANI

NO. OF COPIES	ORGANIZATION
1	SNPE ATTN: L ALLAIN BP2 91710 VERT-LE-PETIT FRANCE
2	MINISTRY OF DEFENSE EXPLOSIVES STORAGE AND TRANSPORTATION COMMITTEE ATTN: J CONNOR M GOULD ROOM 1818/EMPRESS STATE BUILDING LONDON SW6 1TR ENGLAND
1	AGENCY FOR DEFENSE DEVELOPMENT ATTN: S - Y SONG PO BOX 35 YUSEONG TAEJON 305-600 REPUBLIC OF KOREA
1	AUSTRALIAN ORDNANCE COUNCIL DOD ATTN: J GOOLD CANBERRA ACT 2600 AUSTRALIA
1	NATIONAL DEFENSE RESEARCH ESTABLISHMENT FOA ATTN: R FORSEN PO BOX 551 TUMBA S-14725 SWEDEN

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number ARL-TR-645 Date of Report December 1994

2. Date Report Received ______

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) ______

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) ______

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. ______

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) ______

CURRENT ADDRESS	Organization
	Name
	Street or P.O. Box No.
	City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS	Organization
	Name
	Street or P.O. Box No.
	City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)
(DO NOT STAPLE)

DEPARTMENT OF THE ARMY

OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 0001, APG, MD

Postage will be paid by addressee

Director
U.S. Army Research Laboratory
ATTN: AMSRL-OP-AP-L
Aberdeen Proving Ground, MD 21005-5066